Solving Large-Scale Vehicle Routing Problems with Time Windows: The State-of-the-Art

Michel Gendreau
Christos D. Tarantilis

February 2010

CIRRELT-2010-04

Solving Large-Scale Vehicle Routing Problems with Time Windows: The State-of-the-Art

Michel Gendreau[1,*], Christos D. Tarantilis[2]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), and Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

[2] Department of Management Science & Technology, Athens University of Economics & Business, 28, Hydras Street, Athens 113 62, Greece

Abstract. The Vehicle Routing Problem with Time Windows (VRPTW) is a well-known combinatorial optimization problem of high computational complexity. It can be described as the problem of designing least cost routes from a depot to a set of customers of known demand. The routes must be designed such that each customer is visited only once by exactly one vehicle within a given time interval without violating capacity constraints. The goal is to minimize first the total number of vehicles required and second the total travel distance incurred. As research move towards more larger and realistic problems, the trend is towards the development of computationally intelligent algorithms capable of producing high quality solutions with reasonable computational burdens. The purpose of this survey is to discuss the state-of-the-art in the field of advanced heuristics proposed for solving large-scale problem instances. The effort is not only to provide an overview of solution methods, but also to identify the ingredients of success. Computational results, comparing the performance of different algorithms, are reported, while an analysis based on essential attributes, such as efficiency, effectiveness, simplicity and flexibility, is also provided.

Keywords. Algorithms, vehicle scheduling, vehicle routing, time windows.

Acknowledgements. Funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Discovery Grants programs.

---

* Corresponding author: Michel.Gendreau@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec,
Bibliothèque et Archives Canada, 2010

Introduction

Transportation logistics systems are usually large-scale in nature, while their complexity may arise from many different sources, such as customers, vehicles, shipments and physical infrastructure, all interacting in various ways. In addition, due to the continuously increasing collaboration among transportation companies and other logistics partners, problem instances of interest are becoming larger in terms of size and more complex in terms of "new" constraints and objectives. It is common for real life vehicle routing applications (e.g. waste collection, courier and dial-a-ride services and newspaper delivery) to involve the daily service of tens of hundred or even thousand customers.

The Vehicle Routing Problem (VRP) is one of the core operations research and combinatorial optimization problem classes and it has been the object of numerous studies. Solving VRPs optimally or near-optimally for large-scale problem instances is of major interest. With the exception of the Traveling Salesman Problem (TSP), where instances with several thousand nodes can be solved optimally on a regular basis (Gutin and Punnen, 2002), instances of a VRP with more than one hundred customers can be intractably hard to solve optimally and exact methods are confined to limited-sized instances. For this reason, the literature is replete with metaheuristic algorithms, cooperative search methods and hybrid approaches, which stand between heuristics and exact optimization techniques and are capable of producing high quality solutions with computational time burdens reasonable for practical applications. Yet, despite the contributions and the progress the field has seen in recent years, many challenges still stand and new ones are emerging.

Apart from efficiency, scalability and speed of solution procedures are prerequisite attributes for their adaptation within real life decision support systems in which the response times have to be short. Recently, great attention has been devoted on computational experimentation considering large sized data sets for different VRP variants that mimic to some extent real life conditions. Li et al. (2005) and Kytöjoki et al. (2007) reported results for the Capacitated VRP considering data sets with up to 1200 and 24000 customers, respectively. For the Heterogeneous Fleet VRP and the Open VRP, Li et al. (2006, 2007) conducted experiments with more than 480 customers, while for the VRPTW recent algorithmic developments are evaluated on the large-scale problem instances provided by Gehring and Homberger (1999) with a cardinality ranging from 200 to 1000 customers.

The focus of this paper is to provide an overview and discuss the state-of-the-art in the field of advanced heuristics proposed to address large-scale VRPTW instances. The VRPTW is certainly the most studied variant of VRPs and it can be considered the prototype of the so-called "rich" VRPs because time window constraints require sophisticated techniques for constant time feasibility checks (Irnich, 2008). It occurs in several transportation logistics systems and it can be used to model numerous real life applications (Bodin et al., 1983). Typical large-scale VRPTW applications can be found in waste collection (Sahoo et al., 2005), fast food routing (Russell, 1995), school bus routing (Bracca et al., 1994), home delivery and technical-dispatching services (Weigel and Cao, 1999). One may also refer to Golden et al. (2002) for a collection of vehicle routing and scheduling applications from several industries, including solid waste, beverage, food, dairy and newspaper.

The VRPTW can be formally stated as the problem of designing least cost routes from a depot to a set of geographically scattered customers of known demand. The routes must be designed such that each customer is visited only once by exactly one vehicle within a given time interval that models the earliest and the latest times during the day that service can take place. The vehicle must remain at the customer location during the service, and in case the vehicle arrives before the customer is ready to begin the service, it waits. Finally, all feasible routes start and end at the depot, while the accumulated quantities to deliver/collect up to any customer of a route must not exceed the vehicle capacity.

The VRPTW usually has multiple objectives. The goal is to determine the minimum number of vehicle routes and the optimal sequence of customers visited by each vehicle, such that all customers are served and all constraints imposed by vehicle capacity, service times and time windows are satisfied. Typically, a lexicographic (hierarchical) ordering of objectives is followed. That is, the number of routes is first minimized and then, for the same number of vehicle routes, the total travel time or the total travel distance incurred by the fleet of vehicles is minimized. An additional objective might consider minimization of the total time spent (including vehicle's waiting time). In the case of m-VRPTW (fixed fleet size) the first objective is to maximize the total number of customers served. Herein, the focus is on solution approaches proposed for hard time windows following the hierarchical ordering of objectives described above.

The VRPTW can be decomposed into two sub-problems. If the time window constraints are relaxed, it reduces to a bin-packing problem (or set-partitioning), while if the capacity constraints are relaxed, it results in a multiprocessor scheduling problem with some form of sequence-dependent service times, as well as release times and deadlines to account for time windows. The complexity of the VRPTW, which is ***NP***-hard in its general form, stems from the inherent complexity of its underlying constituting elements, since both are ***NP***-complete (Garey and Johnson, 1979). Even the problem of finding a feasible solution for the m-VRPTW is ***NP***-complete (Savelsbergh, 1985).

Early work on the VRPTW dates back to the 1960's. For an overview on early developments, we refer readers to the surveys of Golden and Assad (1986), Desrochers et al. (1988), Desrosiers et al. (1995), and Cordeau et al. (2002). The more recent surveys of Bräysy and Gendreau (2005a,b) and Bräysy et al. (2004a) review the algorithmic developments in the field of construction heuristics, iterative improvement heuristics, metaheuristics and evolutionary algorithms. Since the late 1980s, most solution approaches have been evaluated upon the medium-sized benchmark data sets of Solomon (1987), while recently the focus of most researchers has shifted on large-scale problem instances using the data sets of Gehring and Homberger (1999). Although significant contributions have been made regarding the Solomon's (1987) data sets, no method can consistently re-produce the best-known solutions and few can find the best-known fleet size for all problem instances. On the other hand, the best-known results for the large-scale problem instances have been continuously updated for the last 9 years; however, there is still much room for improvement, especially in terms of effectiveness. Evidently, many solution approaches fail to provide a good compromise between quality and speed, while few score well on other dimensions, such as simplicity and flexibility.

The main contribution of this paper is to discuss the latest research activities in the field of advanced heuristics proposed for solving large-scale VRPTW instances. The main effort is not only to provide an overview of the literature, but also to identify the ingredients of success both in terms of accuracy and speed. Initially, we focus on the design of neighborhoods and we discuss recent advances in the field of neighborhood search methods, since both the structure and the evaluation of neighborhoods may affect significantly the resulting performance. Afterwards, we divide the literature into three categories and we elaborate our analysis for each solution approach based on four attributes: efficiency, effectiveness, simplicity and flexibility, as originally proposed by Cordeau et al. (2002). Apart from the constituting elements, great attention is given to speed up techniques, parallel implementations, integrative and collaborative algorithmic schemes and route elimination strategies. To this end, computational results comparing the performance of different solution approaches are also reported. Finally, the main findings are summarized and conclusions are drawn offering pointers for future research.

1 Neighborhood: Structure and Evaluation

Neighborhood structures and neighborhood search methods are key algorithmic components, since the majority of solution methods for large-scale VRPTW instances are local search-based or make use of local search components. Despite their success, only a small fraction exploits the interrelationships between neighborhood definitions and the associated neighborhood search algorithms. Clearly, the neighborhood size and the speed of evaluation determine to a large extent the resulting efficiency and effectiveness. Ambitious large neighborhoods increase the chances of finding high quality solutions, but they are more expensive to explore. This is why methods for intelligently pruning neighborhoods are important. Proper structural design and the choice of the appropriate search method are ways to achieve this goal.

Finding the best solution in a given neighborhood is itself an optimization problem. This problem can be solved either exactly or heuristically, using direct enumeration schemes (e.g. sequential search and lexicographic search) or indirect optimization algorithms (e.g. branch-and-bound and network optimization algorithms for shortest paths or cycles). Apart from the solution technique, the speed of neighborhood evaluation depends also on the computational effort to determine the cost of a neighboring solution and checking its feasibility. Furthermore, if the search is enumerative there is always the flexibility to terminate the evaluation process whenever one improving neighbor from the set of all improving neighbor solutions has been found. In particular, there are three well-known strategies: the first improvement, the best improvement and the *d*-best improvement. From the average viewpoint these strategies might significantly differ in their efficiency, while the worst-case performance of all strategies is equivalent (Irnich, 2008).

Since there is always the trade-off between the size and the effort needed for searching the entire neighborhood, effective local search implementations for solving large-scale problem instances should essentially utilize neighborhoods of at least manageable size. Below, both classical and newly developed neighborhoods are described. We focus on two major building blocks: the size and the evaluation of neighborhoods. Attention is given first to edge-exchange neighborhoods, edge handling techniques and direct enumerative search methods, while an overview of search methods for large neighborhoods in the context of the VRPTW is also provided.

1.1 Node- and Edge-exchange Neighborhoods

1.1.1 Neighborhood Definition

Edge-exchange neighborhoods are generated by a set of operations, such as segmentation, inversion, permutation and concatenation, made at the level of edges, and typically their size is a polynomial function of the total number of customers (Funke et al., 2005). Apart from the number of deleted and added edges, a common way to manage the size of edge-exchange neighborhoods is to use a set of restricted operators. Most of these operators are structured in such a way that only proper move types are considered during permutation and concatenation, the length of the segments (or paths) is restricted during segmentation and some segments might need to be reversed during inversion operations. Similarly, there are operators that can be used to extend the size of a neighborhood structure. A typical example is the Generalized Insertion operator introduced by Gendreau et al. (1992, 1995), in which a local re-optimization is performed simultaneously with an insertion.

Any intra-route edge-exchange neighborhood can be derived from the *k*-Opt neighborhood (also known as *k*-exchange) that involves the substitution of a set of *k* edges with a set of *k* other edges from a single route. The total number of possible *k*-Opt moves (including proper and non-proper moves) for a given k is $2^{k-1}(k-1)!$, while verifying k-optimality requires $O(n^k)$ operations. Due to the rapid increase of the computational requirements w.r.t k, k is limited to 2 or 3 in most cases (2-Opt and 3-Opt neighborhoods). The size of the 2-Opt neighborhood is quadratic and there is only one proper move type, while the size of 3-Opt is cubic and there are four proper move types. The only 3-Opt move type that does not reverse any segment is the so-called Or-Opt move (Or, 1976), which relocates a segment of consecutive customers of length usually less than or equal to three. The size of the Or-Opt neighborhood is quadratic (if the length of the segment is bounded). Finally, an extension called inverted Or-Opt (IOpt) was introduced by Bräysy (2003) considering both the forward and reserve orientation of the relocated segment.

The k-Opt* neighborhood generalizes the k-Opt one to account for multiple-routes (Potvin et al., 1989). The idea behind the application of a 2-Opt* neighborhood is to swap the end segments between two routes without reversing the order of customers. The size of the 2-Opt* neighborhood is quadratic (Potvin and Rousseau, 1995). Similarly, the Path Insertion (also known as String Relocation) generalizes the Or-Opt and IOpt neighborhoods. More specifically, Path Insertion relocates a segment of at least two consecutive customers from one route to another considering both the forward and reverse orientation. The size of the Path Insertion neighborhood is cubic.

The λ-interchange neighborhood proposed by Osman (1993) can be considered as the generic inter-route node-exchange neighborhood structure. It selects two subsets of customers (with cardinality less than or equal to λ) from two different routes and exchanges them considering all possible insertion positions for both routes, resulting in a neighborhood size $O(n^{4\lambda})$. If it is required that the nodes be inserted in the position of the removed nodes, the size reduces to $O(n^{2\lambda})$. Since the size of λ-interchange neighborhood is relatively large even for small values of λ, in the literature λ rarely exceeds 2 and most researchers adopt small and manageable special structured λ-interchange sub-neighborhoods. Typical examples are the Relocation, the Exchange and the Swap neighborhood of Savelsbergh (1992) and the CROSS-exchange neighborhood structure introduced by Taillard et al. (1997).

The idea behind the Relocation move (also known as Insertion) is the removal of one node from its current position and its insertion into a different position of the same or a different route. The size of the Relocation neighborhood is quadratic. In an Exchange (or Swap) local move, one node is moved from one route to another and a second node is moved from the latter to the former. On the other hand, the CROSS-exchange move involves the exchange between two segments of any length (or less than a predefined length) from two different routes. The size of the CROSS-exchange neighborhood is $O(n^4)$. Finally, several structures have been proposed in the literature that extend the above described neighborhoods, such as inverted CROSS (Bräysy, 2003), GENI-exchange and GENICROSS (Bräysy, 2004b).

1.1.2 Direct Enumeration Search Methods & Acceleration Techniques

Direct enumeration search methods for $O(n^k)$ neighborhoods typically work on a search tree with at least k levels by adding and/or deleting edges and evaluate the results of these operations directly. The latter requires the implicit or explicit construction of neighbors, the computation of cost (or gain compared to the current solution) and the test of whether the neighbor is feasible or not. The goal is to reduce the search effort (perform less than n^k operations) by pruning early in the search, or by finding the local best neighbor as quickly as possible without scanning all neighboring solutions explicitly (Funke et al, 2005). The critical criteria for the reduction of the search space are cost and feasibility, i.e., if some branch of the search tree does not contain any feasible or improving neighbor, it can be pruned. In the literature, there are two main approaches, namely the lexicographic search (Kindervater and Savelsbergh, 1997) and the sequential search (Irnich, 2008) based on feasibility and cost reductions, respectively.

Lexicographic search is based on the systematic way in which segments are built. In particular, during segmentation the k edges for removal are chosen in a lexicographic order (the order in which edges currently appear in the routing plan) using k nested loops. Having fixed the first edge, all other edges selected for removal are chosen sequentially, such that exactly one edge is added or deleted from one segment in every loop. Thus, from one neighbor to the next only the last selected edge is shifted on and on. The latter implies that information from the previous segmentation, permutation, inversion and concatenation operations can be used to check the feasibility of the current move and to prune the search early (Savelsbergh, 1992). In this regard, lexicographic search yields the sufficient conditions to prove that if the current solution is infeasible, the entire following sequence is infeasible, too. Furthermore, if for every segment all necessary routing and scheduling information is efficiently updated and maintained in a set of global variables, then the feasibility check of a move can be made in constant time instead of $O(n)$ (Kindervater and Savelsbergh, 1997).

Sequential search is a gain-oriented search procedure that exploits the cost and cyclic independence of neighborhoods. The notion is to decompose a move into partial moves (whose sum is the overall gain of the move) and to examine all relevant partial moves of the cyclic independent neighborhood recursively. If the sum of a sequence of gains is positive, then there exists a cyclic permutation of these gains, such that every partial sum is positive. The latter can be generalized to restrict the search considering only those branches where the sum of gains of the first $p \leq k$ partial moves has to be greater than pG^*/k, where G^* is a lower bound of the overall gain. Contrary to lexicographic search, the elements are selected w.r.t to a set of so-called neighbor lists sorted in the order of increasing cost. One disadvantage is that feasibility of a move cannot be checked before all elements of the partial moves have been specified (Funke et al., 2005).

So far, most local search based solution methods for large-scale VRPTW instances adopt lexicographic search based approaches for the evaluation of edge-exchange neighborhoods. To this end, a generic implementation of sequential search for the VRPTW has been recently illustrated by Irnich (2008). According to his results on the data sets of Gehring and Homberger (1999), there is a significant speedup when lexicographic search is replaced by sequential search, especially for the long-haul less constrained problem instances. Similar results were observed on a small set of 10 very large-scale problem instances with up to 10,000 customers.

One way to further accelerate sequential search-based approaches is to use restricted candidate lists instead of complete neighbor lists. These candidate lists can be either time-oriented or distance-oriented, i.e., candidate lists that contain for each customer *i* a list of the *l* nearest customers from *i*. A typical paradigm is the granularity concept introduced by Toth and Vigo (2002). During the evaluation process, they assume that the first added edge must belong to a candidate list. The latter contains edges whose length is below a certain threshold that varies during the search process in an effort to intensify or diversify the search. Effective implementations of time-oriented candidate lists for both medium and large-scale VRPTW instances are provided by Ibaraki et al. (2005, 2008) and Hashimoto et al. (2008).

In general terms, time windows are a hard constraint to deal with, since it combines routing with scheduling aspects. Checking the feasibility of a route in a straightforward way (looping over all nodes) requires at least linear time. A typical way is to reduce the computational complexity is to treat segments as nodes (Kindervater and Savelsbergh, 1997). The main idea is to reduce the original graph by contracting entire sequences of nodes into objects (called macronodes) with a limited number of properties summing up all the information needed to evaluate solutions. The new graph is much like the old one, i.e., a time window is associated to each object and a travel time to each edge connecting objects. In the VRPTW literature, most implementations follow similar approaches for testing feasibility. For a formal description of the macronodes approach we refer to Cordone and Wolfler (1996).

Finally, updating mechanisms can be considered as alternative acceleration techniques. The effect of any node- or edge-exchange move is limited to the routes it modifies. Thus, it is actually unnecessary to evaluate all neighbors at each step. It is adequate to keep the feasible improving exchanges and their gain in a data structure. Once a move is executed, all those involving modified routes are removed from the data structure and re-evaluated again, whereas the others need not to be updated, since they remain feasible and alter the objective function by the same amount (Cordone and Wolfler-Calvo, 2001). As mentioned by Repoussis et al. (2009), such updating mechanisms are particularly effective especially for large-scale short-haul problem instances with tight time window constraints.

1.2 Large Neighborhoods

Neighborhoods of exponential size and neighborhoods that are too large to search explicitly (enumerate and evaluate all neighbors) in practice, are generally considered to be *large neighborhoods*. Such large neighborhood structures allow a more thorough exploration of the search space, and the possibility of reaching directly high quality solutions is thus higher. However, a large neighborhood does not necessarily produce a more effective local search-based procedure unless one can search the neighborhood effectively (Ahuja et al., 2000). As mentioned by Gendreau and Potvin (2005), large-size neighborhoods are attractive only if they are coupled with filtering techniques aimed at focusing the search on promising regions, given that a complete evaluation of the entire neighborhood is costly. Typical examples are the cyclic-exchange neighborhood proposed by Thompson and Psaraftis (1993) and the ejection chain concept suggested by Glover (1992).

The cyclic-exchange neighborhood generalizes in a sense the k-exchange one. The main idea is to improve the cost of a set of routes by transferring a small number of demands among routes cyclically. Neighboring solutions are obtained by transferring k single elements among subsets (or routes) of a solution, where k varies from 1 to K (the maximum number of elements). Let a solution containing m routes and ρ denote a cyclic permutation of a subset of $\{1,..., m\}$. The simultaneous transfer of demands from a route r^j to $r^{\rho(j)}$ for each j is a cyclic transfer. If the cyclic permutation has a fixed cardinality b then the b-cyclic k-transfer neighborhood is obtained.

Identifying the optimum cost-decreasing cyclic exchange is computationally intractable. A possible alternative is to decompose the very large neighborhood into a set of partial moves that explicitly map to the decision variables of an auxiliary optimization problem defined on a network. The latter is called an *improvement graph* and can be used to implicitly search the neighborhood via appropriate network optimization algorithms. The improvement graph of interest is defined w.r.t. a feasible partition of S (the set of routes) and is denoted as $G(S)$. Let $S[a_j]$ denote the subset containing the elements a_j. $G(S)$ is a directed graph with n nodes, where each node i corresponds to the element $a_i \in S$. A directed arc $(i; j)$ in $G(S)$ signifies that element a_i leaves its current subset and moves to the subset containing element a_j, that is, the subset $S[a_j]$, and simultaneously the element a_j leaves $S[a_j]$. As such, a cost-decreasing cyclic exchange w.r.t. S corresponds to a negative cost subset-disjoint (directed) cycle in $G(S)$. The problem of finding a negative cost subset-disjoint cycle is ***NP***-complete, and thus, only heuristic methods are adequate. Thompson and Psaraftis (1993) used a polynomial-time approximation to calculate the arc costs and restricted the subsets using small negative cost cycles. A more effective technique for identifying negative cycles is proposed by Ahuja et al. (2000, 2001). Their method modifies the well-known label-correcting algorithm for the shortest-path problem (from one node to all other nodes), where each directed path maintained is a subset-disjoint path. A similar approach considering alternative heuristic search strategies for finding shortest cycles was proposed by Gendreau et al. (2006).

Ejection Chains (EC) are based on the concept of generating compound moves, leading from one solution to another, by linked steps in which changes in selected elements cause other elements to be "ejected from" their current state, position or value assignment (Glover, 1992, 1996). The idea is to move towards intermediate-reference structures before moving to another solution, instead of moving directly from one solution to another. The reference structure resembles the solution structure, but allows violation of certain types of constraints. Using a set of predefined transition rules, moves are generated from feasible solutions to reference structures, from one reference structure to another and back from reference structures to solutions. Clearly, a certain amount of infeasibility is introduced into the initial solution, which has to be "ejected" to end up with a new feasible trial solution. However, the ejection of infeasibility can be delayed by moving to other reference structures first to create a chain effect. As such, at each level (depth) trial solutions are available, while for a given depth the goal is to find the best trial solution observed along chains. Reference structures proposed by Glover (1996) are the Stem and Cycle and its generalization, the Doubly Rooted Stem and Cycle, while Sontrop et al. (2005) proposed a Constrained Doubly Rooted structure for the VRPTW.

A more frequently used type of EC is the so-called node-ejection chains. The notion is to combine series of node-exchanges into a compound move. The latter refers to the removal of a node from one route and the re-insertion of this node into a different route, by first removing another node from the latter. The removal and re-insertion operations are repeated, until the last ejected node is either inserted into the position left empty by the first removed node or it is inserted freely into any position of a route that does not intersect with the chain. Thus, for each root node, a tree is formed in which leaves denote nodes that can be freely inserted into other positions of the same or of different routes assuming an empty intersection with the tree. The main difference between node-ejection chains and cyclic exchanges is that only one node is shifted within a partial move of the former, while the ordering of the nodes in the new route of the latter might be completely different (Funke et al., 2005).

The search for cost-decreasing node-ejection chains can be made via a depth or a breadth search technique, iterating all chains for each root node. Clearly, heuristics can be also employed, using topological parameters to restrict the search, such as maximum depth and width of the tree. In a way similar to the use of cyclic improvement graph (move-composition approach) described earlier, Gendreau et al. (2006) modeled the task of finding the best chain or cycle of ejection/insertion partial moves as a constrained shortest path problem. The latter is solved heuristically using an adaptation of the all-pairs Floyd-Warshall shortest path algorithm along with a priori fixing ordering strategies. On the other hand, Caseau et al. (1999) adopted a domain-dependent Constraint Programming (CP) search approach equipped with the Limited Discrepancy Search (LDS) heuristic pruning technique proposed by Harvey and Ginsberg (1995).

CP is a paradigm for representing and solving a wide variety of problems. Typical examples of search techniques that can be used within CP frameworks are depth-first search for constraint satisfaction and branch-and-bound for optimization. Depth-first search with constraint propagation work as follows. At each node of the search tree, the propagation mechanism removes values from the domains of constrained variables that are inconsistent with other variables. If the propagation mechanism removes all values from any of the variables, then there cannot be a solution in this sub-tree and the search backtracks making a different decision at the previous choice point. Similarly, at each step of a branch-and-bound, a variable is chosen to instantiate and sub-problems are explored in turn with the variable assigned to each of its values. After each assignment constraint propagation occurs. This propagation accounts for all constraints, as well as for a bound on the objective function. During the search process, if the domain of one or more constrained variables becomes empty the search backtracks at the previously made assignment. Readers wishing to learn more about the integration of CP frameworks with local search heuristics are referred to Pesant and Gendreau (1999).

Instead of exploring completely the search tree or truncating the search to a limited number of backtracks, one could use within the CP framework an LDS technique that prunes heuristically the unpromising branches. The idea behind LDS is to explore the search tree in order of increasing number of discrepancies, where a *discrepancy* refers to the assignment to a variable of a value that is not the best according to the value ordering heuristic used in the search (which could be interpreted as the heuristic making a mistake). Therefore LDS can be viewed as exploring the search tree in waves, with each successive wave allowing the heuristic to make more mistakes (Bent and Van Hentenryck, 2004). Thus, wave i explores the solutions that can be reached by assuming that the heuristic made i mistakes. An alternative usage of LDS is to branch only in a limited number of cases to those nodes of the search tree that the heuristic decision is least compelling (Caseau et al., 1999).

Another very popular family of large neighborhood structures is the so-called partially destructive/constructive neighborhoods. The rationale is first to remove a set of customers or segments from the current routing schedule and then re-insert back the removed components using a constructive algorithm. The latter can be either an exact optimization technique (e.g., branch-and-bound) or a simple insertion-based heuristic algorithm (e.g., cheapest insertion). There are four key elements that characterize partially destructive/constructive neighborhood structures: (i) the total number of removed customers, (ii) the corresponding customer selection criteria, (iii) the treatment of the remaining parts of the solution and (iv) the way the removed customers are re-inserted.

Shaw (1998) introduced the concept of Large Neighborhood Search (LNS) within an integrated CP framework. LNS can be interpreted as follows: positions of some customer visits within the current vehicle routing schedule are relaxed and re-optimized in the best possible manner by solving smaller sub-problems. The set of removed customers is selected randomly with a bias towards groups of related customers (geographically close with similar demand and starting time for service). These customers are then re-inserted back to the solution at optimal cost. The re-insertion process is performed using an LDS-based branch-and-bound search technique (branching against the best insertion) with the limit on the bound set at the cost of the solution before the relaxation took place. Finally, during this local search process, if a number of consecutive non-improving attempted moves are observed, the number of customers removed is increased. This gradual expansion of the neighborhood shares many similarities with the Variable Neighborhood Search metaheuristic solution framework of Mladenović and Hansen (1997).

Rousseau et al. (2002) introduced three CP-based LNS neighborhoods, namely LNS-GENI, Naive Ejection Chains (NEC) and SMAll RouTing (SMART). The basic idea behind LNS-GENI is to extend the basic LNS scheme and to account in addition for re-insertion positions between non-consecutive customers. The customers to be removed are chosen randomly with a bias towards customers generating the longest detours. For the re-insertion of customers, a first fail strategy is followed attempting the insertion of the most constrained customers first. On the other hand, NEC is used primarily for removing a particular customer from a route rather than reducing the distance cost. The customers with the larger detour from the target route are selected first for ejection. This choice increased the probability of reducing the travel distance and creates a larger temporal space for the re-insertion of a customer. Finally, SMART removes a set of edges instead of customers forming an incomplete solution. The removed edges can be either selected in a consecutive fashion or randomly with a bias towards long edges.

Earlier, Russell (1995) proposed a neighborhood structure that removes up to four customers and reinserts them using partial enumeration, combined with the parallel insertion heuristic of Potvin and Rousseau (1993). Similar neighborhood structures have been put forward by Bräysy (2003), namely the intra-route O-Opt and the inter-route Insert Related Parallel (IRP). Overall, the above described partially constructive/destructive neighborhoods combined either with insertion-based heuristics or optimization-based techniques are often employed successfully within advanced heuristics for solving both small and large-scale VRPTW instances.

A natural avenue to speed-up neighborhood search algorithms is to introduce spatial and temporal decoupling in an effort to define sub-problems that can be optimized independently and reinserted into an existing solution. Bent and van Hentenryck (2007) proposed a so-called randomized adaptive spatial decoupling (RAND) utilizing the LNS of Shaw (1998). RAND considers spatial decoupling and produces independent feasible sub-problems that do not share customers or vehicles. It views the customer region as a circle and randomly selects a wedge to define the sub-problem. Initially, it collects the vehicles serving customers within the wedge and also considers all other customers served by these vehicles to define the sub-problem. In an effort to produce sub-problems with equal densities of customers two angles are defined for each wedge. The RAND-LNS has been successfully applied on large-scale VRPTW instances. The major advantage of RAND is that it does not depend on the instance data and it is independent from the underlying optimization algorithm. Similarly, spatial decoupling acceleration techniques have been also utilized by Mester et al. (2006).

2 Advanced Heuristics for the VRPTW

The VRPTW has generated substantial research efforts and it is certainly the most well studied vehicle routing and scheduling problem. As research moves toward more realistic sized problems and research for large-scale optimization is receiving more and more attention, the focus of most researchers has shifted to large-scale problem instances. During the last decade, a plethora of well performing solution approaches have appeared in the literature. Overall, the field seems to have reached a certain level of maturity and the literature can be roughly divided into three categories: *metaheuristic algorithms*, *parallel and cooperative search methods* and *hybrid optimization algorithms*. We now examine the methods proposed in each category to solve large-scale VRPTW instances.

2.1 Metaheuristic Algorithms

As stated by Osman and Laporte (1996) "a metaheuristic is formally defined as an iterative generation process which guides and subordinates heuristics by combining intelligently different concepts for exploring and exploiting the search space, while learning strategies are used to structure information in order to find efficiently near-optimal solutions". Tabu Search (TS) (Glover, 1986), Evolutionary Algorithms (EA) (Bäck et al., 1997), Iterated Local Search (ILS) (Lourenço et al., 2002) and Variable Neighborhood Search (VNS) (Mladenović and Hansen, 1997) are typical examples of metaheuristic algorithms successfully adapted and implemented for the VRPTW.

Bräysy (2003) was one of the firsts to propose methods aimed at solving large-scale VRPTW instances. In this paper, he presents a deterministic multi-phase Variable Neighborhood Descent (VND) approach. Initially, a set of solutions is produced using a hybrid construction heuristic similar to the one suggested by Russell (1995). Next, a breadth-first node-ejection chain approach is applied for route elimination, coupled with a particularly effective intelligent reordering insertion mechanism. To this end, a subset of solutions with minimum fleet size is selected and improved via a VND local search scheme. The latter encapsulates four parameterized inter- and intra-route neighborhoods, namely I-CROSS, IRP, I-Opt and O-Opt. A novel feature is the use of twin-variable structures in which the parameter values used to define the neighborhood structures are modified after each successful cycle in addition to the traditional neighborhood change scheme. Finally, a modified objective function (total route duration in particular) is considered for diversification during the local search process, if the search stagnates for a certain period.

This was quickly followed by another paper (Bräysy et al., 2004b) in which was presented a two-phase multi-start approach combined with a Threshold Acceptance (TA) (Dueck and Scheurer, 1990) post-optimization algorithm. As in Bräysy (2003), a cheapest insertion heuristic is used in the first phase to generate initial solutions using different parameter settings, while a so-called Injection Tree (IT) procedure is applied to minimize the fleet size. Compared to the node-ejection chains of Bräysy (2003), the proposed IT considers in addition simultaneous insertions and ejections of multiple customers. Given a subset of high quality solutions, the second phase applies I-CROSS-exchanges according to a steepest descent iterative improvement strategy, followed by a TA post-procedure, which applies the extended I-Opt and GENICROSS neighborhood structures.

In 2006, Pisinger and Ropke (2006) proposed a very different approach that extended the LNS idea to make it adaptive. This Adaptive LNS algorithm utilizes a set of partially destructive/constructive neighborhoods that compete to modify the current solution in an iterative fashion. Two sets of heuristic operators N^- and N^+ are chosen to destroy (remove a number of customers and place them into a request bank) and re-construct the current solution (insert back customers from the request bank into one or more routes). The new provisional solution is accepted w.r.t. a Simulated Annealing (SA) (Kirkpatrick et al., 1983) criterion, while an adaptive layer stochastically controls the selection of operators on the basis of a roulette wheel selection scheme. The more an operator has contributed to the search process (by providing improvements to the solution), the higher is the probability that it will be chosen in the future. For diversification, a noise function is used within all operators. Finally, in terms of fleet size minimization, an additional stage prior the standard Adaptive LNS is suggested. Given an initial solution, a route is removed and all its customers are placed into the request bank. Next, Adaptive LNS is applied, and if a feasible solution that serves all customers is encountered, the procedure is repeated with a smaller fleet size until no feasible solution can be obtained.

Later, Ibaraki et al. (2008) presented an ILS algorithm capable of dealing with instances with piecewise linear convex time penalty functions, assuming soft time windows for the customers, as in Ibaraki et al. (2005). Given the visiting sequence of customers served by a vehicle route, a

Dynamic Programming (DP) procedure is used to determine the optimal service times, such that the penalized objective function is minimized. The proposed DP is used during the evaluation of neighboring solutions. The complexity for convex time penalty functions is $O(\delta_k \log \delta_k)$ for each vehicle *k*, while the time to evaluate a neighboring solution is $O(\log \delta_{max})$. As in Ibaraki et al. (2005), the ILS local search applies restricted 1-Opt, 2-Opt, 2-Opt*, Path Insertion and 1-CROSS-exchange neighborhood structures within a VND local search framework. Distance and time oriented neighbor-lists are also used to accelerate the neighborhood evaluation process.

A similar ILS implementation was proposed by Hashimoto et al. (2008) for large-scale Time-Dependent VRPTW instances. Given an initial solution, an iterative improvement local search heuristic is applied until no further improvement can be obtained, while the overall ILS framework iterates from solutions generated by perturbing the local optimum solutions obtained previously during the search. Their local search heuristic uses 2-Opt*, CROSS-Exchange and Or-Opt neighborhood structures, which are restricted in terms of size. The complexity of the modified DP algorithm is the same as in Ibaraki et al. (2008), despite the fact that is capable of dealing with time-dependent travel times. An interesting feature of their implementation is the use of information from past DP recursions, combined also with a filtering method, in order to reduce the search effort during the evaluation of neighborhoods.

A multi-parametric (1+1)-evolution strategy (ES) was proposed by Mester et al. (2006). At first, five solutions are generated via a hybrid parallel savings heuristic algorithm and the best among them forms the initial population. During the evolution process, a new solution is produced via a so-called Remove-Insert multi-parametric mutation operator and the parent is replaced if an improvement is observed. As a result, after a number of generations a set of modified solutions is obtained. Three operators are used for the removal of customers (purely random removals, removal of one customer from each route, and random ejections from rings generated from two circles centered on the depot with random radiuses), while a cheapest insertion heuristic is applied for re-insertion. Each offspring is further improved using Or-Opt, Exchange, 2-Opt local moves within a parameterized dynamic environment, called “adaptive variable neighborhood” (AVN). In an effort to prune the neighboring space, a strategy for selecting neighboring routes, called “dichotomous route combinations” (DRC), is also used to take advantage of the geographical division and topology of the vehicle routes.

Mester and Bräysy (2005) proposed a hybrid metaheuristic algorithm. They combined Guided Local Search (GLS) with an ES, similar to the one described above, into a two-phase interactive approach. In the first phase, GLS is used to regulate a composite local search. Initially, a set of solutions is generated using a hybrid parallel savings heuristic with different parameter settings. During the construction process, Relocate and Exchange local moves are applied cyclically in a periodic fashion. The best solution of the set is further improved using the SA route minimization procedure of Bent and Van Hentenryck (2004). The GLS step considers long arcs as the feature to penalize, on the basis of the Relocate and Exchange neighborhoods. A particular feature is that the neighborhoods are restricted to the set of geographically close routes w.r.t. the current penalized arc, these are called "penalty variable neighborhoods" (PVN). If no further improvement can be obtained via the GLS, an (1+1)-evolution strategy is triggered. After a number of generations, the best individual is used as the starting point for the next restart of GLS. The oscillation between GLS and ES is repeated until some termination conditions are met.

Hoshino et al. (2007) developed a local search heuristic controlled by chaotic dynamics exploiting principles of neural networks. The chaotic search applies Exchange and Relocate local moves. To control these local moves a set of neurons (each neuron corresponds to a customer and a local move) is defined. At each neuron, a gain, a refractory and a mutual reflect are assigned. The gain effect is related to the distance savings obtained by the neuron, the refractory effect inhibits the firing of a neuron that has been just fired (memory effect with an exponential decay), and the mutual effect adjusts a firing ratio to all neurons. The overall framework works as follows. Each neuron is fired at a particular time and if some necessary conditions are met, the local move that corresponds to this neuron is executed. The neurons are updated asynchronously, while a single iteration is performed whenever all neurons are updated. Finally, the route elimination heuristic of Bräysy (2003) is applied periodically for further improvement.

An Arc-Guided Evolutionary Algorithm was proposed recently by Repoussis et al. (2009). The latter manipulates a population of individuals on the basis of an (μ+λ)-ES, utilizing a discrete arc-based representation combined with a binary vector of strategy parameters. Following a strictly generational course of evolution and a deterministic selection scheme of survivors, offspring are produced via mutation out of arcs extracted from parents. The mutation operator is based on the ruin-and-recreate principle, while a multi-parent recombination operator enables the self-adaptation of the strategy parameters based on the appearance frequency of each arc and the diversity of the population. Each resulting offspring is further improved via a route elimination procedure and a hybrid Guided Tabu Search algorithm. Both approaches utilize the basic TS local search framework to drive the search process. However, the primary objective of the former is to reduce the number of vehicles using a hierarchical ordering of objectives similar to Bent and van Hentenryck (2004), while the latter explores the solution space on the basis of a GLS algorithm in an effort to reduce the total distance. For the evaluation of the neighboring space, a lexicographic search approach is followed coupled with neighborhood updating mechanisms for further acceleration.

Hashimoto and Yagiura (2008) suggested a Path Relinking solution approach. Initially, a reference set of randomly generated solutions is build, while an iterative improvement local search using the 2-Opt*, CROSS-exchange and Or-Opt neighborhoods is applied. During relinking, the starting and guiding solutions are selected randomly, while a random perturbation is applied with some probability to the guiding solution for diversification. Among the solutions generated in the relinking sequence, local search is applied to a subset of local optima. In an effort to reduce the size and speed up the neighborhood exploration, time-oriented neighbor lists and segment length restrictions are also considered. During the local search process, infeasible solutions are allowed, while the amount of violation is penalized. Apart from capacity and temporal infeasibilities, frequency based penalties are also employed to account for customers that appear often in infeasible routes. To this end, an adaptive mechanism is used to control the weights of these penalties. For the computation of the penalized objective function in constant time, traveling times for particular paths are stored and the neighborhood is searched in a specific order, as in Ibaraki et al. (2005).

Finally, Nagata et al. (2010) proposed a two-stage Edge-Assembly Memetic Algorithm (EAMA). During the vehicle minimization stage, a population of individuals is generated and the route elimination heuristic of Nagata and Bräysy (2009a) is applied. The main feature of the subsequent distance minimization stage is the so-called edge assembly crossover (EAX) operator (Nagata and Bräysy, 2009b). The key idea of the latter is to generate a fixed number of offspring solutions by combining directed edges from two parent individuals A and B selected randomly, such that the capacity and time window constraints are not considered and the number of routes is not changed. A subsequent local search-based repair procedure tries to restore the feasibility of the infeasible offspring. It uses a generalized cost function consisting of the distance traveled and two penalty components imposed on constraint violations. For evaluating time window violations, a new penalty function is proposed: when a late arrival occurs at a given customer of the route, time is "pushed backwards" in order to arrive on time (like using a time machine). By doing so, the schedule for the subsequent customers of the route is no longer disturbed by the time delay at the customer under consideration. Each time the "time machine" is used to arrive earlier at a given customer, the associated penalty value (push backward time) is cumulated. Feasible solutions are further improved via a local search procedure, which uses the 2-opt*, Or-Exchange, Relocation, and Exchange neighborhood structures. For the selection of survivors in each generation, an elitist selection scheme is adopted.

2.2 Parallel and Cooperative Search methods

Cooperative search is a category of parallel algorithms, in which several algorithms run in parallel and share information to solve an optimization problem; it is a parallelization strategy for search algorithms, where parallelism is obtained by concurrently executing several search programs (Crainic et al., 1997; Crainic and Toulouse, 2002). From a different viewpoint, teamwork hybridization represents cooperative optimization models consisting of many parallel cooperating agents, each carrying out a search in the solution space (Talbi, 2002). Three forms of parallelism can be applied to metaheuristics as proposed by Crainic and Toulouse (2002): (i) low-level (operational) parallelism aims solely to speed-up computations by accelerating steps of an algorithm, without any attempt at achieving a better exploration; (ii) search space decomposition parallelism, where each algorithm searches for a sub-solution in a sub-space by partitioning the set of decision variables into disjoint subsets, and (iii) multi-search threads, in which several algorithms perform multiple explorations for a solution in the whole space. In the latter case, each concurrent thread may or may not be executed by the same method. These methods may start from the same or different solutions and they can communicate during the search. Thus, depending on the operating scheme, these approaches can be either purely independent search methods or cooperative multi-thread search strategies. Finally, communication can be made synchronously or asynchronously and it can be event-driven or executed at predetermined time instances.

Le Bouthillier and Crainic (2005) presented a cooperative search method in which several search threads communicate through asynchronous exchanges of information using a pool of feasible solutions, called a solution warehouse. Communications are initiated only by individual threads that have access to the data sent by other threads. However, no broadcasting takes place. The methods involved in the cooperation scheme consist of simple construction and local search heuristics, including Genetic Algorithms (GA), TS algorithms (similar to Cordeau et al., 2001, and Gendreau et al., 1994) and a post-elimination procedure. The initial population is generated using different construction heuristics. Local optimum solutions are sent to the warehouse from the TS algorithms, while initial solutions are received either to restart or to diversify the search. On the other hand, the two GAs use the solution warehouse as their population. Parents are selected randomly and a probabilistic mutation is performed. Two crossover operators are considered, the order crossover (OX) and the edge recombination (ER). When required, a repair procedure restores feasibility by re-ordering or re-routing customers with infeasible time windows. Each offspring is send to the solution warehouse. Finally, the post-optimization procedure consists of a node-ejection chain and several local search improvement heuristics. The latter is applied to all solutions maintained within the warehouse.

Later, Le Bouthillier et al. (2005) modified the cooperative search method of Le Bouthillier and Crainic (2005). The proposed method is a *guided parallel cooperative search method* and it is based on a central memory structure called data warehouse that contains solutions and pattern information. A particular feature is the introduction of a sophisticated knowledge extraction mechanism that is used to guide each search method (global search). The suggested mechanism extracts knowledge from the information exchanged among search threads in an effort to identify promising or unexplored regions of the solution space (coordination). For this purpose, the mechanism incorporates a pattern-identification procedure that is used to fix or prohibit specific solution attributes (arcs in particular) for part of the search. The definition of patterns is based on the inclusion of arcs within the solutions, while the solution warehouse is divided into subsets according to the quality of solutions (i.e., elite, average and worst). An arc with a high frequency in a given subset of solutions signals that the search methods participating into the cooperation have often produced solutions that use this particular arc. Each pattern contains a subset of arcs. Frequent pattern considers arcs with high appearance frequency, while the opposite is assumed for infrequent patterns. To this end, patterns can be derived from specific sub-populations, while the rate of appearance of a specific in-pattern among them can be also defined indicating whether an in-pattern is promising or not. These promising and unpromising patterns are used to constrain for some time the search processes, and thus induce global intensification or diversification phases. From the implementation viewpoint, initially two phases of diversification are launched to broaden the search, followed by two intensification phases that mainly focus on promising regions.

Gehring and Homberger (1999) developed earlier a parallel two-phase evolutionary algorithm combined with TS. In the first phase, the ES proposed by Homberger and Gehring (1999) is performed following an $(1,\lambda)$ evolution scheme to minimize fleet size, while in the second phase TS is applied for distance minimization. During the evolution process, the recombination of individuals is omitted, while one offspring is generated per parent via mutation. The latter uses Or-Opt, 2-Opt* and Exchange local moves, while the mutation code indicates for each individual the times a randomly selected local move is applied. For the evaluation of individuals a customized fitness function is utilized. It consists of two indices in a lexicographic order: the number of customers served by the smallest vehicle and the sum of minimal delays of these customers (relative difficulty of relocation).The two-phase solution approach is parallelized assuming cooperative autonomy (i.e., autonomous sequential search threads cooperate through the exchange of solutions), while each independent search thread is performed with different configuration settings.

Later, Gehring and Homberger (2001, 2002) modified this solution approach by moving to an $(\mu;\lambda)$-ES and a different fitness function considering also capacity infeasibilities. As in Homberger and Gehring (1999), the fitness function is based on four criteria; the total number of vehicles, the number of customers of the smallest route, the minimal delay and the total distance traveled in a lexicographic order. In addition to the minimal delay, the property of the so-called "caused overload" is also introduced. The latter indicates the extent of overload of a particular route, if the customer with the smallest demand of the smallest route is relocated to that route. These two fitness values are used to configure differently the concurrently executed search threads. However, the final selection of individuals is made according to the traditional VRPTW objective function. Other variants considered new termination criteria (i.e., ES is terminated once the minimum number of vehicles is reached and TS terminates if no further improvement can be obtained for a number of iterations). Finally, a single-thread implementation of this method can be found in Homberger and Gehring (2005) with a different move-generation mechanism, termination conditions and fitness function.

2.3 Hybrid Optimization Algorithms

During the last years, several authors have been started developing hybrid optimization algorithms that combine heuristic and exact techniques. A natural avenue for the cooperation between exact and heuristic methods is either to design a heuristic method to improve the search strategy of an exact approach or to use an exact optimization technique to explore large neighborhoods within local search algorithms. Clearly, LNS-based approaches fall into this category of cooperation since they rely on exact techniques to build partial solutions that are then used to define the search space for the associated local search framework, while the results obtained by the local search may also provide feedback to refine bounds or columns and so on.

Bent and Van Hentenryck (2004) developed a two-phase LNS-based hybrid optimization algorithm. The first phase consists of a SA route elimination approach, while the second phase mainly employs a modified LNS algorithm. The main feature of the suggested SA is the utilization of a specialized hierarchical objective function that contains three components: the first is related to the total number of vehicles, the second favors solutions containing routes with many customers and routes with few customers over solutions where customers are distributed more evenly among the routes, while the third accounts for the minimal delay. From the implementation viewpoint, SA is restarted several times from the best encountered solution. In each inner iteration, although neighborhoods and customers are randomly selected, the associated sub-neighborhoods are exhaustively examined. If the best neighbor improves the best solution found, it is accepted in a fashion similar to the TS aspiration criterion. Otherwise, a neighbor is randomly selected and accepted if the current solution is improved or if the probabilistic acceptance criterion of SA is met. On the other hand, the proposed LNS implementation considers a number of modifications compared to Shaw (1997), including among others a restart strategy and tighter lower bounds.

A hybrid LNS method that utilizes a heuristic branch-and-price technique was proposed by Prescott-Gagnon et al. (2009). The LNS-based solution approach is equipped with four partially destructive/constructive neighborhoods. The first destruction operator is called Proximity (Shaw, 1998) and it is based on a measure of the spatio-temporal proximity among customers. The second is called SMART (SMAll RouTing) (Rousseau et al., 2002) and ensures that more isolated customers are disconnected from their routes. The third operator is called Longest Detour

(Rousseau et al., 2002) and selects customers at random with a bias towards those yielding the largest detours. The fourth selects all customers that are currently visited within a given time period. For the selection of operators, a roulette wheel selection scheme similar to the one proposed by Pisinger and Ropke (2006) is utilized. During the solution reconstruction process, the partially fixed problem is modeled as a set partitioning problem where each variable corresponds to a vehicle route. The associated problem is solved via a heuristic branch-and-price, where the column generation subproblem can be interpreted as an elementary shortest path with resource constraints. This subproblem is solved by a TS heuristic, which performs a limited number of feasible moves on each route associated with a basic variable of the master problem. The procedure terminates when the branch-and-price cannot generate any negative reduced cost columns. If the solution is fractional, an integer solution is obtained by exploring a search tree using a depth-first strategy, without backtracking. Finally, fleet size is minimized at the begin of the algorithm through a special stage that puts emphasis on vehicle reduction and sets an upper bound on the number of vehicles that can be used. This lower bound is lowered whenever a feasible solution covering all customers is found.

Finally, Lim and Zhang (2007) proposed a two-phase multi-start algorithm. The first phase consists of two procedures for generating initial solutions and for minimizing the fleet size, while the second phase is devoted to distance minimization. The initial solutions are generated via squeaky-wheel optimization (SWO) (Joslin and Clements, 1999) using Solomon's (1987) I1 insertion-based heuristic. At each iteration of the latter, the solution generated is analyzed and penalties are increased by a particular amount that is proportional to the number of customers served per route and inversely proportional to the unutilized capacity. These new penalties are then used to guide the next construction. Afterwards, the route elimination procedure is applied. More specifically, a data structure called Ejection Pool (EP) is used to hold temporarily unassigned customers. At each iteration, a customer from the EP is selected and inserted into the existing set of routes. The target route and the insertion position are determined w.r.t. a weighted combination of distance and service delay. If the insertion is not feasible, "relevant" customers are kicked out sequentially (starting from those with the largest kick saving) and added to the EP until the route becomes feasible. Kick saving considers the distance reduction obtained and the cost needed for the re-insertion into another route w.r.t. the minimal delay. To this end, an iterative improvement hill-climbing local search procedure is employed. Initially, two sequential iterative improvement heuristics are applied interchangeably until no further improvement can be obtained. The first performs an intra-route improvement, using Exchange, Relocate, 2-Opt and Or-opt neighborhoods. When a local optimum is reached an enumeration based E-Opt move is applied. E-Opt move selects a consecutive segment of customer and determines the optimum ordering using branch-and-bound for a predefined maximum length of segments. The second heuristic operates on pairs of routes using 2-Opt* and CROSS-exchanges. If both heuristics fail to improve the current solution, a so-called Generalized Ejection Chain (GEC) is applied. The search for cost-decreasing chains is performed heuristically using the very large neighborhood search algorithm of Ahuja et al. (2000, 2001), while if GEC fails to find a better solution, the best found solution is perturbed.

2.4 Research Trends and Common Issues

In agreement with a general trend observed in the literature for solving hard combinatorial optimization problems, most advanced heuristics described above do not purely follow the particular principles or concepts from a single solution framework. Indeed, more and more solutions approaches aim at exploiting the strengths of different methods and suggest the cooperation between several heuristics or metaheuristics and between heuristics and exact optimization techniques. Interaction can take place either at low-level, by designing new methods that combine various algorithmic principles, or at high-level, by developing multi-agent architectures such as cooperative search methods. In the former case, one deals with the functional composition of a single method and particular functions of a solution framework are replaced by other methods (e.g., neighborhood search within a metaheuristic algorithm being performed using exact optimization techniques). In the latter case, the different methods are self-contained and are executed either in a sequential (pipeline) or in a parallel (cooperative) fashion.

The emergence of such hybrid solution approaches offers new opportunities for combining the strengths and alleviating the weaknesses of different frameworks into more powerful and flexible search methods that perform an extensive and intelligent exploration of the search space. Towards the design of advanced heuristics, Gendreau and Potvin (2005) proposed a unified framework based on several common algorithmic components that emerged from algorithmic principles and recent implementations. Using these common components both the basic implementation schemes and other recent developments can be described, showing the unification towards integration of these particular components. For a recent review and taxonomy on hybrid solution approaches between heuristics and exact optimization techniques, we refer readers to Jourdan et al. (2009).

The majority of metaheuristic algorithms proposed for solving large-scale VRPTW instances are stochastic, except from those of Bräysy (2003) and Hoshino et al. (2007), while in several occasions many of them utilize additional randomization features for the better exploration of the search space. Although, the accuracy of metaheuristic algorithms is adequately high, they are often complex to implement and calibrate requiring finely tuned parameters. However, metaheuristic algorithms remain so far the only viable approach for solving efficiently and time effectively large-scale problem instances. Most implementations utilized either simple edge-exchange neighborhood structures or large neighborhoods equipped with heuristic construction operators. To this end, most approaches utilize lexicographic search and feasibility based speed up techniques, while some use neighbor lists and spatial decomposition mechanisms to accelerate the neighborhood search process.

Among parallel and cooperative search methods, Gehring and Homberger (1999, 2001), Le Bouthillier and Crainic (2005) and Le Bouthillier et al. (2005) adopt parallel explorations strategies, where several search threads run concurrently. Carefully designed parallel solution approaches have several advantages, such as solving a larger problem in a given time (scale up), generating results in a shorter time (speed up), finding higher quality solutions in a given time, improving convergence behavior and so on (Crainic et al., 1997). Important issues in such schemes are mostly related to the communications among the search threads. In Gehring and Homberger (1999, 2001), the parallel search is coupled with adaptive memories. Under this scheme, threads cooperate weakly by exchanging only good solutions. On the other hand, Le Bouthillier and Crainic (2005) and Le Bouthillier et al. (2005) adopt more integrated communication schemes by sharing both solutions and global information to guide the search threads.

The common ground among all hybrid optimization methods proposed for solving large-scale VRPTW instances is the use and exploitation of exact optimization techniques for the exploration and evaluation of large neighborhoods. According to the taxonomy of Jourdan et al. (2009), all these solution approaches fall into the category of low-level teamwork hybrids. Although, the use of large neighborhoods combined with sophisticated optimization techniques seems to be important for obtaining high quality solutions, the resulting simplicity and flexibility is reduced. Furthermore, in many cases these solution approaches are more context-dependent compared to others, while they may also require large amounts of computational time.

Finally, another common characteristic of most approaches is the use of route elimination procedures as independent algorithmic components. Bräysy (2003) and Bräysy et al. (2004b) use modified node-ejection chains equipped with specialized reordering mechanisms. Bent and Van Hentenryck (2004), Mester and Bräysy (2005), Repoussis et al. (2009) and Gehring and Homberger (1999, 2002) utilize customized objective functions combined with local search procedures. Pisinger and Ropke (2006) and Lim and Zhang (2007) use holding lists of customers and allow the search to enter infeasible regions. Similarly, Ibaraki et al. (2008), Hashimoto and Yagiura (2008) and Hashimoto et al. (2008) utilize penalized objective functions and accept infeasible solutions during the course of the search. Although, it is advantageous to deal with feasible solutions as proposed by Repoussis et al. (2009), it seems that the approaches of Hashimoto and Yagiura (2008) and Lim and Zhang (2007) are currently the best among those for route elimination.

Recently, a very powerful route minimization heuristic that outperforms all existing approaches was proposed by Nagata and Bräysy (2009b). They use an ejection pool of customers, diversification mechanisms to guide the ejection operations along with squeeze and perturb procedures. Initially, a route from the current solution is randomly selected. Its customers are temporarily removed from the solution and are transferred to the so-called ejection pool (EP). Then, a repeated attempt is made to insert the customers from the EP into the existing set of routes, avoiding capacity and time window constraint violations. In the case where are no feasible insertion positions for the customer selected from the EP, other customers are ejected from one of the existing routes to create a feasible insertion position for the selected customer and the EP is updated accordingly. The above described ejection – insertion mechanism iterates until all customers of the EP are served in a feasible manner.

3 Comparative Analysis

3.1 Benchmark Data Sets

Empirical analysis is the most common way for the evaluation of solution approaches. It involves testing the solution approach across a wide range of problem instances to get a performance indication. For the VRPTW, empirical analysis is typically based on the results obtained w.r.t. Solomon's (1987) 56 benchmark problem instances. These instances contain 100 customers, a central depot, vehicle capacity constraints, time windows on the time of delivery and a total route duration restriction. For testing scaling issues, the 300 large-scale problem instances of Gehring and Homberger (1999) are also considered.

The data set of Solomon (1987) consists of six different classes, namely R1, C1, RC1, R2, C2 and RC2. Each class contains between 8 and 12 100-node problems over a service area defined on a 100x100 grid. The Cartesian coordinates of customers in classes R1 and R2 are randomly generated with a uniform distribution, while classes C1 and C2 have clustered customers. Finally, classes RC1 and RC2 contain semi-clustered customers, i.e., a combination of both clustered and randomly (uniformly) distributed customers. All problems of a particular class have the same customer locations and the same vehicle capacities, but the percentage of customers with time window constraints (i.e., 25%, 50%, 75% and 100% time window density) differs. Classes R1, C1 and RC1 have tight time windows, short scheduling horizons and a homogeneous vehicle fleet of vehicles of capacity equal to 200 units, thus allowing only few customers per route (short-haul). Classes R2, C2, RC2 have longer scheduling horizons and vehicle capacities ranging from 100 to 1000 units, which allows the service of a larger number of customers per route (long-haul). Finally, travel times and distances are given by the Euclidean distance between customer's locations. The features of Gehring and Homberger (1999) benchmark data set are similar. It consists of 300 problem instances divided into 5 groups, namely G02, G04, G06, G08 and G10. Each group maintains the characteristics and the structure of Solomon's (1987) data set described above. However, each group has a much larger cardinality w.r.t. the size of the set of customers, i.e., 200, 400, 600, 800, or 1000 customers.

The experimental results obtained for the VRPTW are ranked according to a hierarchical objective function. The primary objective is to minimize the total number of vehicles, and for a given fleet size, the secondary objective is to minimize the total traveled distance. Therefore a solution requiring fewer routes is always better than a solution with more routes, regardless of the total traveled distance. However, these two objectives can be either conflicting or complementary, since the reduction of the total number of vehicles may either increase or reduce respectively the total traveling distance. Thus, comparisons among different approaches are valid only if the above hierarchy of objectives is followed and the total number of vehicles obtained is the same.

It is important to note that the solutions provided for problem instances R1_2_1 and C1_2_8 by Bräysy (2003) and Mester and Bräysy (2005), for instance R1_4_1 by Bräysy (2003), and for instance R1_8_1 by Bent and Van Hentenryck (2004) are infeasible w.r.t. the number of vehicle routes (Nagata et al., 2010).

3.2 Effectiveness and Efficiency Analysis

The comparison among different algorithmic approaches in terms of effectiveness and efficiency is a difficult task, since a number of issues must be first addressed for the competition to be fair and objective. In particular, a measure has to be defined for comparing computational CPU running times from different machines. Following the suggestions made by other authors, a relative (estimated) speed is derived for each machine w.r.t. a Pentium IV 2.8 GHz using the performance indicators of Dongarra (2008) and other empirical metrics. Another important issue is that most authors report only the best results obtained during multiple executions. Since most solution approaches are stochastic, each execution for the same problem instance would provide different results. Therefore, only average results based on multiple executions would be a good basis for the comparison of non-deterministic methods. Herein, a common basis for all approaches is provided, while the reported computational time is multiplied by the number of runs (different experiments) and the number of available processors.

Tables 1, 2, 3, 4, 5 and 6 summarize the results obtained w.r.t. Solomon's (1987) and Gehring and Homberger's (1999) benchmark data sets. Each table is divided into three parts. The first column of the first part refers to the different classes R1, R2, C1, C2, RC1 and RC2 respectively. For each class, the mean number of vehicles (MNV) and mean distance traveled (MTD) are reported. The second part contains the lines CNV and CTD, which indicate the cumulative number of vehicles and the cumulative distance traveled over all problem instances. The last part describes the computer, the number of processors, the number of independent runs and the average CPU time in minutes as reported by authors, along with the resulting estimated speed and time. Finally, the first line lists the authors using the following abbreviations: **HY** for Hashimoto and Yagiura (2008), **GH99** for Gehring and Homberger (1999), **GH02** for Gehring and Homberger (2001, 2002), **HG** for Homberger and Gehring (2005), **BHD** for Bräysy et al. (2004b), **MB** for Mester and Bräysy (2005), **MBD** for Mester et al. (2006), **LC** for Le Bouthillier and Crainic (2005), **LCK** for Le Bouthillier et al. (2005), **BVH** for Bent and Van Hentenryck (2004), **I** for Ibaraki et al. (2008), **LZ** for Lim and Zhang (2007), **PR** for Pisinger and Ropke (2006), **DPR** for Prescott-Gagnon et al. (2009), **HYI** for Hashimoto et al. (2008), **RTI** for Repoussis et al. (2009), **NBD** for Nagata et al. (2010), **HKI** for Hoshino et al. (2007) and **B** for Bräysy (2003). In all tables, boldface entries indicate best known results.

According to Table 1, the solution approaches of Prescott-Gagnon et al. (2009), Le Bouthillier et al. (2005), Bent and Van Hentenryck (2004), Lim and Zhang (2007), Pisinger and Ropke (2006), Hashimoto and Yagiura (2008), Hashimoto et al. (2008), Repoussis et al. (2009), Nagata et al. (2010), and Bräysy (2003) are able to obtain the lowest known CNV. Le Bouthillier et al. (2005) and Homberger and Gehring (2005) produce the best results for classes R1 and R2, while Prescott-Gagnon et al. (2009) and Nagata et al. (2010) report the best results for classes RC1 and RC2 respectively. For the clustered classes C1 and C2, most approaches perform equally well. As for the minimization of distance traveled, Repoussis et al. (2009) and Nagata et al. (2010) currently report the best results. However, the differences compared to other methods are relatively small. Finally, in terms of computational time consumption the solution approach of Bräysy (2003) seems to be the most time efficient for the data sets of Solomon (1987).

The observations for the large-scale data sets of Gehring and Homberger (1999) differ significantly from the previous ones as one moves from 200 to 1000 customers. According to Table 2, for the 200-customer problem instances the approaches of Nagata et al. (2010) and Prescott-Gagnon et al. (2009) perform best in terms of CTD among those with the lowest CNV, while the most time efficient is the approach of Gehring and Homberger (1999). More specifically, Nagata et al. (2010) provide the best results for most classes, while Prescott-Gagnon et al. (2009) report the best for class C2. Overall, the differences in terms of CNV are small and several methods are able to obtain the lowest known values.

For the 400-customer problem instances (see Table 3), Nagata et al. (2010) and Repoussis et al. (2009) obtain the lowest CNV, while the approach of Nagata et al. (2010) is the most effective and yields the best mean results for all classes. Low CNV values are also obtained by Lim and Zhang (2007), Prescott-Gagnon et al. (2009), Hashimoto and Yagiura (2008) and Hashimoto et al. (2008).

The results for problem instances with 600 customers are similar (see Table 4). The approach of Repoussis et al. (2009) performs best producing the lowest known CNV, followed by Nagata et al. (2010). High quality solutions are also produced by Lim and Zhang (2007), Hashimoto and Yagiura (2008) and Prescott-Gagnon et al. (2009). Overall, Nagata et al. (2010) report the best MNV and MTD for all classes, except for class C1.

According to Table 5, for problem instances with 800 customers the approach of Hashimoto and Yagiura (2008) is the most effective, produces the lowest CNV and yields the best results for class C2. On the other hand, Nagata et al. (2010) report the best mean values for classes R1, R2, C1 and RC2, while Lim and Zhang (2007) obtain the best for class RC1. Finally, for the 1000-customer problem instances (see Table 6), the approach of Hashimoto and Yagiura (2008) produces the lowest CNV values and obtains the best results for classes C1 and C2. The approach of Lim and Zhang (2007) obtains the same MNV, but lower MTD values for class RC1. For all remaining classes, Nagata et al. (2010) reports the best mean values.

On the basis of the above observations, the solution approaches of Nagata et al. (2010), Repoussis et al. (2009) and Hashimoto and Yagiura (2008) are the most effective and produce for most groups of problems the lowest known CNV values. Among the remaining approaches, low CNV values and very good results for some classes are also obtained by Ibaraki et al. (2008), Pisinger and Ropke (2006), Gehring and Homberger (1999), Mester and Bräysy (2005), Lim and Zhang (2007), Prescott-Gagnon et al. (2009) and Le Bouthillier et al. (2005). In general terms, most approaches are able to obtain low MNV values for classes R1, R2 and C2, while the gap between the best and the worst performing solution approaches increases as moving towards larger problem instances especially for classes RC1, RC2 and C1.

Cumulative results for all solution approaches are provided in Table 7. This table displays the cumulative number of vehicles (CNV), the cumulative distance traveled (CTD) and the cumulative computational time (CCT) consumption in minutes for each group of problem instances. The CCT has been calculated according to the estimated mean average CPU Time for all problem instances of each group. Furthermore, the last column provides the aggregated results over all groups.

Table 1: Effectiveness analysis on the 100-customer data sets of Solomon (1987)

Class	HY	GH99	GH02	HG	BHD	MBD	LC	LCK	BVH	DPR	HYI	I	LZ	PR	RTI	NBD	B
R1	11.92	12.42	12.00	12.08	12.00	12.00	**12.08**	11.92	11.92	11.92	11.92	12.08	11.92	11.92	11.92	11.92	11.92
	1216.70	1198.00	1217.57	1211.67	1214.69	1208.18	**1209.19**	1214.20	1213.25	1210.34	1213.18	1212.09	1213.61	1212.39	1210.82	1210.34	1222.12
R2	2.73	2.82	2.73	**2.73**	2.73	2.73	2.73	2.73	2.73	2.73	2.73	2.73	2.73	2.73	2.73	2.73	2.73
	961.28	947.00	961.20	**950.72**	960.44	954.09	963.62	954.32	966.37	955.74	955.61	960.95	961.05	957.72	952.67	951.03	975.12
C1	**10.00**	10.00	10.00	10.00	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**
	828.38	829.00	828.63	828.45	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**
C2	**3.00**	3.00	3.00	3.00	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	589.86	590.00	590.33	589.96	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**
RC1	11.50	11.88	11.50	11.50	11.50	11.50	11.50	11.50	11.50	**11.50**	11.50	11.50	11.50	11.50	11.50	11.50	11.50
	1384.17	1356.00	1395.13	1395.93	1389.20	1387.12	1389.22	1385.30	1384.22	**1384.16**	1384.25	1391.46	1385.56	1385.78	1384.30	1384.16	1389.58
RC2	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25	**3.25**	3.25
	1138.37	1140.00	1139.37	1135.09	1124.14	1119.70	1143.70	1129.43	1141.24	1119.44	1120.50	1127.00	1121.82	1123.49	1119.72	**1119.24**	1128.38
CNV	405	415	406	408	406	406	407	405	405	405	405	407	405	405	405	**405**	405
CTD	57484	56942	57641	57422	57422	56812	57412	57360	57567	57240	57282	57437	57368	57332	57216	**57187**	57710
Computer	X2,8G	P200M	P400M	P400M	A700M	P800M	P850M	P850M	SU10	O2,3G	P2,8G	P2,8G	P2,8G	P3G	P3G	O2,4G	P200M
Processors	1	4	4	1	1	1	5	5	1	1	1	1	1	1	1	1	1
CPU	17	5	15.1	17.5	2.7	43.8	12	12	120	30	16.7	16.7	93.2	2.4	17.9	5	82.5
Runs	1	1	5	5	30	1	1	1	5	5	3	1	1	10	3	5	1
≈Speed	1.26	0.03	0.06	0.06	0.21	0.12	0.14	0.14	0.16	1.41	1.00	1.00	1.00	1.07	1.07	1.45	0.03
≈Time	21.48	0.53	17.20	4.98	17.22	5.32	8.43	8.43	94.76	211.85	50.10	16.70	93.20	25.77	57.66	36.26	2.19

Table 2: Effectiveness analysis on the 200-customer data sets of Gehring and Homberger (1999)

Class	HY	GH99	GH02	HG	BHD	MB	MBD	LC	LCK	RTI	DPR	HYI	I	LZ	HKI	PR	NBD	B
R1	18.20 3632.85	18.20 3705.00	18.20 3855.03	18.20 3890.06	18.20 3718.30	18.20 3618.68	18.20 3659.10	18.20 3676.95	18.20 3615.06	18.20 3640.11	18.20 3615.69	18.20 3690.34	18.20 3655.24	18.20 3639.60	18.20 3899.25	18.20 3631.23	**18.20** **3612.36**	18.10 3821.43
R2	4.00 2967.02	4.00 3055.00	4.00 3032.49	4.10 3059.78	4.00 3014.28	4.00 2942.92	4.00 2960.50	4.00 2986.01	4.00 2969.90	4.00 2941.99	4.00 2937.67	4.00 2943.88	4.00 2958.56	4.00 2950.09	4.20 3087.07	4.00 2949.37	**4.00** **2929.41**	4.00 3045.29
C1	18.90 2718.68	18.90 2782.00	18.90 2842.08	19.00 2836.66	18.90 2749.83	18.80 2717.21	18.90 2719.00	18.90 2743.66	18.90 2736.84	18.90 2721.90	18.90 2718.77	18.90 2721.94	18.90 2734.42	18.90 2726.11	19.10 2765.37	18.90 2721.52	**18.90** **2718.41**	18.90 2778.8
C2	6.00 1833.12	6.00 1846.00	6.00 1856.99	6.00 1898.44	6.00 1842.65	6.00 1833.57	6.00 1834.10	6.00 1836.10	6.00 1833.91	6.00 1833.36	**6.00** **1831.59**	6.00 1835.96	6.00 1833.37	6.00 1834.24	6.00 1909.98	6.00 1832.94	6.00 1831.64	6.00 1842.43
RC1	18.00 3196.86	18.00 3555.00	18.10 3674.91	18.10 3734.32	18.00 3329.62	18.00 3221.34	18.00 3276.00	18.00 3449.71	18.00 3255.97	18.00 3224.63	18.00 3192.56	18.00 3345.01	18.00 3275.38	18.00 3205.51	18.50 3446.61	18.00 3212.28	**18.00** **3178.68**	18.00 3508.07
RC2	4.30 2572.55	4.30 2675.00	4.40 2671.34	4.50 2640.94	4.40 2585.89	4.40 2519.79	4.40 2548.70	4.30 2613.75	4.30 2584.18	4.30 2554.33	4.30 2559.32	4.30 2564.68	4.30 2576.12	4.30 2574.10	4.90 2657.9	4.30 2556.87	**4.30** **2536.22**	4.40 2628.36
CNV	694	694	696	699	695	694	695	694	694	694	694	694	694	694	709	694	**694**	694
CTD	169070	176180	179328	180602	172406	168573	169968	173061	169958	169163	168556	171018	170331	169296	177662	169042	**168067**	176244
Computer	X2,8G	P200M	P400M	P400M	A700M	P2G	P800M	P850M	P850M	P3G	O2,3G	P2,8G	P2,8G	P2,8G	C2D2G	P3G	O2,4G	P200M
Processor	1	4	4	1	1	1	1	5	5	1	1	1	1	1	1	1	1	1
CPU	33	10	2.1	1.6	2.4	8	54.4	10	10	90	53	33.3	33.3	93.2	11	7.7	4.1	18
Runs	1	1	3	3	3	1	1	1	1	1	5	3	1	1	1	10	5	1
≈Speed	1.26	0.03	0.06	0.06	0.21	0.68	0.12	0.14	0.14	1.07	1.41	1.00	1.00	1.00	0.90	1.07	1.45	0.03
≈Time	41.69	1.06	1.44	0.27	1.53	5.47	6.61	7.02	7.02	96.63	374.26	99.90	33.30	93.20	9.94	82.67	29.73	0.48

Table 3: Effectiveness analysis on the 400-customer data sets of Gehring and Homberger (1999)

Class	HY	GH99	GH02	HG	BHD	MB	MBD	LC	LCK	RTI	DPR	HYI	I	LZ	HKI	PR	NBD	B
R1	36.40	36.30	36.40	36.40	36.40	36.30	36.30	36.50	36.40	36.40	36.40	36.40	36.40	36.40	36.60	36.40	**36.40**	36.20
	8544.80	8925.00	9478.22	9547.86	8692.17	8530.03	8648.20	8839.28	8607.97	8514.11	8420.52	8998.63	8788.54	8489.53	8977.48	8540.04	**8403.24**	9154.50
R2	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.10	8.00	**8.00**	8.00
	6262.43	6502.00	6650.28	6683.53	6382.63	6209.94	6317.70	6437.68	6302.08	6258.82	6213.48	6258.00	6251.54	6271.57	6768.18	6241.72	**6148.57**	6547.87
C1	37.60	38.00	38.00	38.10	37.90	37.90	38.00	37.90	37.90	37.60	37.60	37.60	37.60	37.60	38.30	37.60	**37.60**	38.00
	7213.21	7584.00	7855.82	7921.19	7230.48	7148.27	7182.60	7447.09	7223.07	7273.90	7182.75	7444.06	7302.50	7229.04	7331.56	7290.16	**7175.72**	7321.68
C2	11.80	12.00	12.00	12.00	12.00	12.00	12.00	12.00	12.00	11.70	11.90	11.80	11.80	11.70	12.10	12.00	**11.70**	12.00
	3910.11	3935.00	3940.19	4049.71	3894.48	3840.85	3862.70	3940.87	3862.66	3941.70	3874.58	3982.50	3985.21	3942.93	4104.68	3844.69	**3899.00**	3922.71
RC1	36.00	36.10	36.10	36.10	36.00	36.00	36.10	36.00	36.00	36.00	36.00	36.00	36.00	36.00	36.60	36.00	**36.00**	36.10
	8017.89	8763.00	9294.99	9296.75	8305.55	8066.44	8192.20	8652.01	8267.81	8088.46	7940.65	8572.11	8471.85	8005.25	8386.48	8069.30	**7922.23**	8628.74
RC2	8.50	8.60	8.80	9.20	8.90	8.80	8.90	8.60	8.60	8.40	8.60	8.50	8.60	8.50	9.50	8.50	**8.40**	8.70
	5326.87	5518.00	5629.43	5609.88	5407.87	5243.06	5258.80	5511.22	5397.54	5516.59	5269.09	5355.59	5328.84	5431.15	5576.36	5335.09	**5297.96**	5633.28
CNV	1383	1390	1392	1397	1391	1389	1390	1390	1389	1381	1385	1383	1384	1382	1412	1385	**1381**	1390
CTD	392507	412270	428489	431089	399132	390386	394818	408281	396611	395936	389011	406109	401285	393695	411447	393210	**388466**	412088
Computer	X2,8G	P200M	P400M	P400M	A700M	P2G	P800M	P850M	P850M	P3G	O2,3G	P2,8G	P2,8G	P2,8G	C2D2G	P3G	O2,4G	P200M
Processors	1	4	4	1	1	1	1	5	5	1	1	1	1	1	1	1	1	1
CPU	67	20	7.1	5.1	7.9	17	238.5	20	20	180	89	66.6	66.6	295.9	35	15.8	16.2	98.8
Runs	1	1	3	3	3	1	1	1	1	1	5	3	1	1	1	5	5	1
≈Speed	1.26	0.03	0.06	0.06	0.21	0.68	0.12	0.14	0.14	1.07	1.41	1.00	1.00	1.00	0.90	1.07	1.45	0.03
≈Time	84.65	2.13	4.85	0.87	5.04	11.62	28.97	14.05	14.05	193.26	628.47	199.80	66.60	295.90	31.62	84.82	117.47	2.63

Table 4: Effectiveness analysis on the 600-customer data sets of Gehring and Homberger (1999)

Class	HY	GH99	GH02	HG	BHD	MB	LC	LCK	RTI	DPR	HYI	I	LZ	HKI	NBD	PR
R1	54.50	54.50	54.50	54.50	54.50	54.50	54.80	54.80	54.50	54.50	54.50	54.50	54.50	55.30	**54.50**	54.50
	18523.42	20854.00	21864.47	21605.60	19081.18	18358.68	19869.82	18698.37	18781.79	18252.13	20363.15	19963.56	18381.28	18933.87	**18186.24**	18888.52
R2	11.00	11.00	11.00	11.00	11.00	11.00	11.20	11.20	11.00	11.00	11.00	11.00	11.00	11.10	**11.00**	11.00
	12678.99	13335.00	13656.15	13682.21	13054.83	12703.52	13093.97	12989.35	12804.60	12808.59	13047.18	12496.54	12847.31	13753.05	**12330.49**	12619.26
C1	57.40	57.90	57.70	57.90	57.80	57.80	57.90	57.70	**57.30**	57.40	57.50	57.50	57.40	58.00	57.40	57.50
	14163.51	14792.00	14817.25	15086.01	14165.90	14003.09	14205.58	14166.80	**14236.86**	14106.03	14296.96	14128.87	14103.61	14224.83	14067.34	14065.89
C2	17.40	17.90	17.80	18.00	18.00	17.80	17.90	17.90	17.40	17.50	17.40	17.40	17.40	18.40	**17.40**	17.50
	7678.49	7787.00	7889.96	7897.59	7528.73	7455.83	7743.92	7582.61	7729.80	7632.37	7960.14	7991.70	7725.86	7864.15	**7605.07**	7801.29
RC1	55.00	55.10	55.00	55.20	55.00	55.00	55.20	55.20	55.00	55.00	55.00	55.00	55.00	55.50	**55.00**	55.00
	16352.56	18411.00	19114.02	19108.14	16994.22	16418.63	17678.13	16643.27	16767.72	16266.14	17764.33	17395.51	16274.17	17189.74	**16183.95**	16594.94
RC2	11.50	11.80	11.90	12.20	12.10	12.10	11.80	11.80	11.40	11.70	11.50	11.60	11.50	12.80	**11.40**	11.60
	10841.22	11522.00	11670.29	11649.75	11212.36	10677.46	11034.71	10868.94	11311.81	10990.85	11315.28	10743.03	10935.91	11209.08	**10586.14**	10777.12
CNV	2068	2082	2079	2088	2084	2082	2088	2086	**2066**	2071	2069	2070	2068	2111	2067	2071
CTD	800982	867010	890121	890293	820372	796172	836261	809493	**816326**	800797	847470	827192	802681	831747	789592	807470
Computer	X2,8G	P200M	P400M	P400M	A700M	P2G	P850M	P850M	P3G	O2,3G	P2,8G	P2,8G	P2,8G	C2D2G	O2,4G	P3G
Processors	1	4	4	1	1	1	5	5	1	1	1	1	1	1	1	1
CPU	100	30	12.9	10.3	16.2	40	30	30	270	105	100	100	646.9	61	25.3	18.3
Runs	1	1	3	3	3	1	1	1	1	5	3	1	1	1	5	5
≈Speed	1.26	0.03	0.06	0.06	0.21	0.68	0.14	0.14	1.07	1.41	1.00	1.00	1.00	0.90	1.45	1.07
≈Time	126.35	3.19	8.82	1.76	10.33	27.33	21.07	21.07	289.89	741.46	300.00	100.00	646.90	55.12	183.46	98.24

Table 5: Effectiveness analysis on the 800-customer data sets of Gehring and Homberger (1999)

Class	HY	GH99	GH02	HG	BHD	MB	LC	LCK	RTI	DPR	HYI	I	LZ	HKI	NBD	PR
R1	72.80	72.80	72.80	72.80	72.80	72.80	73.10	72.80	72.80	72.80	72.80	72.80	72.80	73.10	**72.80**	72.80
	31978.15	34586.00	34653.88	34976.51	32748.06	31918.47	33552.40	32290.48	32734.57	31797.42	34095.04	33413.41	31755.57	33134.66	**31492.81**	32316.79
R2	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.00	15.10	**15.00**	15.00
	20383.13	21697.00	21672.85	22055.78	21170.15	20295.28	21157.56	20765.88	20618.21	20651.81	20810.51	20174.32	20601.22	22232.77	**19914.97**	20353.51
C1	75.20	76.70	76.10	76.70	76.30	76.20	76.30	76.20	75.20	75.40	75.60	75.80	75.40	76.70	**75.20**	75.60
	25220.58	26528.00	26936.68	26563.59	25170.88	25132.27	25668.82	25612.47	25911.44	25093.38	25915.59	25337.93	25026.42	25242.8	**25151.83**	25193.13
C2	**23.30**	24.00	23.70	24.10	24.20	23.70	24.10	24.00	23.40	23.50	23.40	23.50	23.40	24.40	23.40	23.70
	11689.00	12451.00	11847.92	12018.64	11648.92	11352.29	11985.11	11393.80	11835.72	11569.39	11942.54	11726.25	11598.81	12141.46	11447.27	11725.46
RC1	72.00	72.40	72.30	72.50	73.00	73.00	72.30	72.30	72.00	72.00	72.30	72.40	**72.00**	73.20	72.00	73.00
	31343.54	38509.00	40532.35	38070.24	30005.95	30731.07	37722.62	33971.63	33795.61	33170.01	34358.45	35296.29	**31267.84**	30206.51	31278.28	29478.30
RC2	15.40	16.10	16.10	16.20	16.30	15.80	15.80	15.80	15.50	15.80	15.60	15.80	15.60	16.90	**15.40**	15.70
	16828.93	17741.00	17941.23	17980.04	17686.65	16729.18	17441.60	17202.08	17536.54	16852.38	17173.59	16665.08	16992.79	17995.79	**16484.31**	16761.95
CNV	**2737**	2770	2760	2773	2776	2765	2766	2761	2739	2745	2747	2753	2742	2794	2738	2758
CTD	**1367971**	1515120	1535849	1516648	1384306	1361586	1475281	1412363	1424321	1391344	1442957	1426133	1372427	1409540	1357695	1358291
Computer	X2,8G	P200M	P400M	P400M	A700M	P2G	P850M	P850M	P3G	O2,3G	P2,8G	P2,8G	P2,8G	C2D2G	O2,4G	P3G
Processors	1	4	4	1	1	1	5	5	1	1	1	1	1	1	1	1
CPU	133	40	23.2	18.2	26.2	145	40	40	360	129	133.3	133.3	1269.4	96	27.6	22.7
Runs	1	1	3	3	3	1	1	1	1	5	3	1	1	1	5	5
≈Speed	1.26	0.03	0.06	0.06	0.21	0.68	0.14	0.14	1.07	1.41	1.00	1.00	1.00	0.90	1.45	1.07
≈Time	168.04	4.25	15.85	3.11	16.71	99.09	28.09	28.09	386.51	910.93	399.90	133.30	1269.40	86.74	200.14	121.86

Table 6: Effectiveness analysis on the 1000-customer data sets of Gehring and Homberger (1999)

Class	**HY**	**GH99**	**GH02**	**HG**	**BHD**	**MB**	**LC**	**LCK**	**RTI**	**DPR**	**HYI**	**I**	**LZ**	**HKI**	**NBD**	**PR**
R1	91.90	91.90	91.90	91.90	92.10	92.10	92.20	92.00	91.90	91.90	91.90	91.90	91.90	92.90	**91.90**	92.20
	49650.82	57186.00	58069.61	57072.15	50025.64	49281.48	55176.95	51847.22	51414.26	49702.32	54149.50	54909.63	48827.23	49994.85	**48369.71**	50751.25
R2	19.00	19.00	19.00	19.00	19.00	19.00	19.20	19.00	19.00	19.00	19.00	19.00	19.00	19.10	**19.00**	19.00
	29806.86	31930.00	31873.62	32320.68	31458.23	29860.32	30919.77	30441.05	30804.79	30495.26	30626.04	29589.71	30164.60	32972.23	**29003.42**	29780.82
C1	**94.00**	96.00	95.40	96.10	95.80	95.10	95.30	95.10	94.20	94.30	94.40	94.60	94.40	96.30	94.10	94.60
	42157.55	43273.00	43392.59	43524.95	42086.77	41569.67	43283.92	42403.21	43111.60	41783.27	43066.89	16679.76	41699.32	42387.35	41748.60	41877.00
C2	**28.80**	30.20	29.70	29.90	30.60	29.70	29.90	29.60	29.30	29.50	29.40	29.50	29.30	30.30	29.10	29.70
	17152.20	17570.00	17574.72	17566.99	17035.88	16639.54	17443.50	17164.51	16810.22	16657.06	16822.82	16679.76	16589.74	17341.96	16534.36	16840.37
RC1	90.00	90.00	90.10	90.10	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	**90.00**	90.90	90.00	90.00
	45539.11	50668.00	50950.14	51337.25	46736.92	45396.41	49711.36	46118.08	46753.61	45574.11	49378.71	48768.87	**44818.54**	46794.47	44860.60	46752.15
RC2	18.30	19.00	18.50	18.90	19.00	18.70	18.50	18.50	18.40	18.50	18.30	18.40	18.30	19.80	**18.30**	18.30
	24696.91	27012.00	27175.98	27059.89	25994.12	25063.51	26001.11	25390.40	25588.52	25470.33	26428.81	24667.92	25064.88	27442.97	**24055.31**	25090.88
CNV	**3420**	3461	3446	3459	3465	3446	3451	3442	3428	2745	3430	3434	3429	3493	3424	3438
CTD	**2085125**	2276390	2290367	2288819	2133376	2078110	2225366	2133645	2144830	2096823	2204728	2169452	2071643	2169338	2045720	2110925
Computer	X2,8G	P200M	P400M	P400M	A700M	P2G	P850M	P850M	P3G	O2,3G	P2,8G	P2,8G	P2,8G	C2D2G	O2,4G	P3G
Processors	1	4	4	1	1	1	5	5	1	1	1	1	1	1	1	1
CPU	167	50	30.1	30.7	39.6	600	50	50	450	162	166.7	166.7	1865.4	119	35.3	26.6
Runs	1	1	3	3	3	1	1	1	1	5	3	1	1	1	5	5
≈Speed	1.26	0.03	0.06	0.06	0.21	0.68	0.14	0.14	1.07	1.41	1.00	1.00	1.00	0.90	1.45	1.07
≈Time	211.00	5.32	20.57	5.24	25.26	410.02	35.12	35.12	483.14	1143.96	500.10	166.70	1865.40	107.52	255.97	142.80

Table 7: Cumulative results for the benchmark data sets of Gehring and Homberger (1999)

Reference	Cumulative	200	400	600	800	1000	Aggregated
HY	CNV	694	1383	2068	2737	3420	10302
	CTD	169070	392507	800982	1367971	2085125	4815655
	CCT	2502	5079	7581	10083	12660	37904
GH99	CNV	694.0	1390	2082	2770	3461	10397
	CTD	176180	412270	867010	1515120	2276390	5246970
	CCT	64	128	191	255	319	957
GH02	CNV	696	1392	2079	2760	3446	10373
	CTD	179328	428489	890121	1535849	2290367	5324154
	CCT	86	291	529	951	1234	3092
HG	CNV	699	1397	2088	2773	3459	10416
	CTD	180602	431089	890293	1516648	2288819	5307451
	CCT	16	52	106	187	315	676
BHD	CNV	695	1391	2084	2776	3465	10411
	CTD	172406	399132	820372	1384306	2133376	4909592
	CCT	92	302	620	1003	1515	3532
MB	CNV	694	1389	2082	2765	3446	10376
	CTD	168573	390386	796172	1361586	2078110	4794827
	CCT	328	697	1640	5945	24601	33212
MBD	CNV	695	1390	-	-	-	-
	CTD	169968	394818	-	-	-	-
	CCT	397	1738	-	-	-	-
LC	CNV	694	1390	2088	2766	3451	10389
	CTD	173061	408281	836261	1475281	2225366	5118250
	CCT	421	843	1264	1686	2107	6321
LCK	CNV	694	1389	2086	2761	3442	10372
	CTD	169958	396611	809493	1412363	2133645	4922071
	CCT	421	843	1264	1686	2107	6321
BVH	CNV	697	1393	2091	2778	3468	10427
	CTD	171715	410112	858040	1469790	2266959	5176616
	CCT	-	-	-	-	-	-
I	CNV	694	1384	2070	2750	3431	10329
	CTD	170331	401285	827192	1426133	2155374	4980315
	CCT	1998	3996	6000	7998	10002	29994
LZ	CNV	694	1382	2068	2742	3429	10315
	CTD	169296	393695	802681	1372427	2071643	4809742
	CCT	5592	17754	38814	76164	111924	250248
PR	CNV	694	1385	2071	2758	3438	10346
	CTD	169042	393210	807470	1358291	2110925	4838938
	CCT	4960	5089	5894	7312	8568	31823
B	CNV	694	1390	-	-	-	-
	CTD	176244	412088	-	-	-	-
	CCT	29	158	-	-	-	-
HKI	CNV	709	1412	2111	2794	3493	10519
	CTD	177662	411447	831747	1409540	2169338	4999734
	CCT	596	1897	3307	5205	6451	17457
DPR	CNV	694	1385	2071	2745	3432	10327
	CTD	168556	389011	800797	1391344	2096823	4846531
	CCT	22456	37708	44487	54656	68638	227945
HYI	CNV	694	1383	2069	2747	3430	10323
	CTD	171018	406109	847470	1442957	2204728	5072282
	CCT	5994	11988	18000	23994	30006	89982
NBD	CNV	694	1381	2067	2738	3424	10304
	CTD	168067	388466	789592	1357695	2045720	4749540
	CCT	1784	7048	11008	12008	15358	47206
RTI	CNV	694	1381	2066	2739	3428	10308
	CTD	169163	395936	816326	1434321	2144830	4960576
	CCT	5798	11595	17393	23191	28989	86966

Apart from effectiveness, it is important to study the resulting time efficiency of each solution approach. The longer a heuristic is running the better is the final output. Therefore, a compromise is needed in order for an algorithm to produce high quality solutions consuming reasonable amounts of time. As suggested by Bräysy and Gendreau (2005b), the tradeoff between running time and solution quality can be viewed in terms of a multi-objective optimization in which the two objectives are balanced. Thus, performance measures can be plotted in two dimensions. The first corresponds to the running time and the second to the solution quality. In this two-dimensional space, points with the better values on both dimensions are said to be Pareto optimal.

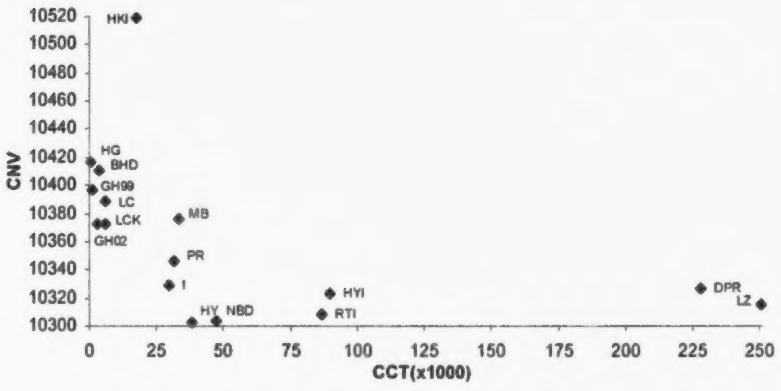

Figure 1: Efficiency Analysis - Large-scale data sets

Figure 1 illustrates the two-dimensional space w.r.t. the CNV and the CCT of each approach according to the values of Table 7. CNV is selected since it is the primary objective and it provides a direct measure for comparing different approaches. The closer is a point to the lower left corner (low CNV using the least CPU time), the better is the associated solution approach. Clearly, the most effective and efficient approaches are those of Nagata et al. (2010) and Hashimoto and Yagiura (2008). The most time consuming approach is one proposed by Lim and Zhang (2007), with a CCT almost 10 times larger compared to the most effective approach. One may argue that the approaches of Ibaraki et al. (2008), Pisinger and Ropke (2006), Repoussis et al. (2009) and Hashimoto et al. (2008) provide a good compromise between speed and accuracy. Finally, it must be mentioned that the estimated average CPU time consumptions provides only an indication and cannot be used for direct quantitative comparisons.

3.3 Simplicity and Flexibility Analysis

In order to arrive at some final conclusions that are relevant for real life applications, the criteria of simplicity and flexibility must be also considered. Simplicity refers to the ease of implementation. Although a minimum of complexity should be expected, simple codes, preferably short and self-contained, stand a better chance to be adopted (Cordeau et al., 2002). Another issue related to simplicity is the number of parameters incorporated. Algorithms that contain too many parameters are difficult to understand and to tune. On the other hand, flexibility is related to the ability of an algorithm to accommodate various side constraints. A flexible algorithmic approach must be able to handle these changes, without requiring much effort. Thus, algorithmic flexibility is in part achieved through the simplicity of the design. Furthermore, flexibility is directly related to the robustness of an algorithm. A robust algorithm must not overly be sensitive to the differences in problem characteristics and should perform consistently over a variety of problem instances.

Table 8: Assessment of advanced heuristics for large-scale VRPTWs

Reference	Effectiveness	Efficiency	Simplicity	Flexibility
HY	Very High	High	High	Very High
GH99	Medium	Very High	Medium-High	High
GH02	Medium-High	Very High	Medium-High	High
HG	Medium-Low	Very High	Medium-High	High
BHD	Medium-Low	Very High	Very High	High
MB	Medium-High	High	High	High
MBD	Medium-High	Medium	High	High
LC	Medium	Very High	Medium	High
LCK	Medium-High	Very High	Medium	High
BVH	Medium	-	Medium	High
I	High	High	Medium	High
LZ	High	Low	Medium-Low	High
RTI	Very High	Medium	Medium-High	High
HYI	High	Medium	Medium	High
PR	High	High	Very High	Very High
B	Medium-High	High	High	High
HKI	Very Low	Medium	High	High
NBD	Very High	High	Medium-High	High
DPR	High	Low	Medium-Low	High

Table 8 summarizes the findings of our analysis. Most approaches obtain medium to high accuracy with medium to very high speed, while in terms of simplicity and flexibility most score above medium. However, implementations that utilize fairly simple structured neighborhoods and search techniques gain additional credits. We also paid attention to the number and the settings of parameters. Finally, it should be noted that the approach of Pisinger and Ropke (2006) has been applied to a wide variety of different VRP variants. In conclusion, among all the approaches proposed for solving large-scale VRPTW instances, those of Hashimoto and Yagiura (2008), Nagata et al. (2010), Repoussis et al. (2009), Pisinger and Ropke (2006) and Ibaraki et al. (2008) score well on all dimensions. The most flexible approaches are those proposed by Pisinger and Ropke (2006) and Hashimoto and Yagiura (2008). Simple structured approaches with relatively few parameters are those of Nagata et al. (2010), Hashimoto and Yagiura (2008), Pisinger and Ropke (2006) and Bräysy et al. (2004b).

4 Conclusions

The VRPTW is a well-known *NP*-hard combinatorial optimization problem occurring in several transportation logistics systems. During the last decade, a large collection of successful and highly efficient advanced heuristics has been proposed for the VRPTW. Until recently most research efforts were targeting relatively small problem instances. However, as research moves towards more realistically sized problems, research on large-scale instances has received much more attention. Indeed, recent solution approaches are constantly producing more accurate results, with significant contributions to the best known solutions. However, there is still ample room for improvement both in terms of effectiveness and efficiency. Furthermore, it is fair to say that many approaches fail to provide a good compromise between quality and running time, while few score well on other dimensions, such as simplicity and flexibility.

Evidently, research for the VRPTW must focus on more effective, simpler and faster solution methods capable of performing an extensive, but also intelligent, exploration of the search space. Real world needs solution methods that are fast, easy to understand, flexible, accurate and robust in terms of consistent performance across different problem solving scenarios. To this end, metaheuristic algorithms are currently the best available option for solving large-scale VRPTW instances. However, hybrid optimization algorithms combining heuristics and exact optimization techniques are likely to match the effectiveness of existing metaheuristic algorithms in the future, since these solution approaches have not yet been fully exploited. The latter is also true considering parallel and cooperative search methods in an effort to get the full performance from the widely available multi-core CPUs.

A promising line of research for metaheuristic implementations is towards the development of more effective and efficient neighborhood search methods and speed-up techniques. Several ideas have been put forward, such as the granularity principle of Toth and Vigo (2003) and the sequential search of Irnich et al. (2005). Furthermore, the adaptation and development of spatial and temporal decompositions, as in Bent and van Hentenryck (2007), seem to be very promising. Another research direction worth pursing is towards the full utilization of CP frameworks within local search heuristics (Pesant and Gendreau, 1999). In CP, every computation is driven by constraints, thus giving them an active role. A significant benefit of CP techniques is that side constraints can be incorporated with comparative ease. Finally, it is interesting to further investigate pattern identification mechanisms, similar to those suggested by Le Bouthillier et al. (2005), and to invest more in analogous guided cooperative search methods.

References

R.K. Ahuja, J.B. Orlin and D. Sharma, "Very Large-scale Neighborhood Search", *International Transactions in Operations Research* 7, 301-317 (2000).

R.K. Ahuja, J.B. Orlin and D. Sharma, "Multi-exchange Neighborhood Structures for the Capacitated Minimum Spanning Tree Problem", *Mathematical Programming* 91, 71-97(2001).

T. Bäck, U. Hammel and H.-P. Schwefel, "Evolutionary Computation: Comments on the History and Current State", *IEEE Transactions in Evolutionary Computation* 1(1), 3-17, 1997.

R. Bent and van Hentenryck P., "A Two-stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows", *Transportation Science* 38, 515-530 (2004).

R. Bent and van Hentenryck P., "Randomized Adaptive Spatial decoupling for large-scale vehicle routing with time windows", In *Proceedings of the Twenty-Second Conference on Artificial Intelligence of the American Association for Artificial Intelligence*, 2007, Vancouver, BC., pp.173-178.

L. Bodin, B. Golden, A. Assad and M. Ball, "Routing and Scheduling of Vehicles and Crews: The State-of-the-Art", *Computers & Operations Research* 10, 62-212 (1983).

J. Bracca, J. Bramel and D. Simchi-Levi, "A Computerized Approach to the New York City School Bus Routing Problem", *IEE Transactions* 29, 693-702 (1994).

O. Bräysy, "A Reactive Variable Neighborhood Search for the Vehicle Routing Problem with Time Windows", *INFORMS Journal on Computing* 15, 347-368 (2003).

O. Bräysy, W. Dullaert and M. Gendreau, "Evolutionary Algorithms for the Vehicle Routing Problem with Time Windows", *Journal of Heuristics* 10, 587-611 (2004a).

O. Bräysy and M. Gendreau, "Tabu Search Heuristics for the Vehicle Routing Problem with Time Windows", *TOP* 10(2), 211-237 (2002).

O. Bräysy and M. Gendreau, "Vehicle Routing Problem with Time Windows Part I: Route Construction and Local Search Algorithms", *Transportation Science* 39, 104-118 (2005a).

O. Bräysy and M. Gendreau, "Vehicle Routing Problem with Time Windows Part II: Metaheuristics", *Transportation Science* 39, 119-139 (2005b).

O. Bräysy, G. Hasle and W. Dullaert, "A Multi-start Local Search Algorithm for the Vehicle Routing Problem with Time Windows", *European Journal of Operational Research* 159, 586-605 (2004b).

Y. Caseau, F. Laburthe and G. Silverstein, "A Meta-Heuristic Factory for Vehicle Routing Problems", in *Lecture Notes in Computer Science*, J. Jaffar (eds), Vol. 1713, 144-159, Constraint Programming - CP99, Springer, Berlin, 1999.

G. Clarke and J.W. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", *Operations Research* 12, 568-581 (1964).

J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon and F. Soumis, "The VRP with Time Windows", In P. Toth and D. Vigo (eds), *The Vehicle Routing Problem*, 157-193, SIAM Monographs on Discrete Mathematics and Applications, 2002.

J.-F. Cordeau, M. Gendreau, G. Laporte, J.-Y. Potvin and F. Semet, "A guide to vehicle routing heuristics", *Journal of the Operational Research Society* 53, 512-522 (2002).

J.-F. Cordeau, G. Laporte and A. Mercier, "A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows", *Journal of the Operational Research Society* 52, 928-936 (2001).

R. Cordone and R. Wolfler-Calvo "A Heuristic for the Vehicle Routing Problem with Time Windows", *Journal of Heuristics* 7, 107-129 (2001).

T.G. Crainic and M. Toulouse, "Parallel Strategies for Meta-heuristics", in *Handbook of Metaheuristics*, F. Glover and G.A. Kochenberger (eds), 475-513, International Series in Operations Research and Management Science, Kluwer Academic Publishers, Norwell, 2002.

T.G. Crainic, M. Toulouse and M. Gendreau, "Toward a Taxonomy of Parallel Tabu Search Heuristics", *INFORMS Journal on Computing* 9(1), 61-72 (1997).

M. Desrochers, J.K. Lenstra, M.W.P. Savelsbergh and F. Soumis, "Vehicle Routing with Time Windows: Optimization and Approximation", In B. Golden and A. Assad (eds), 65-84, *Vehicle Routing: Methods and Studies*, Elsevier Science Publishers, Amsterdam, 1988.

J. Desrosiers, Y. Dumas, M.M. Solomon and F. Soumis, "Time Constrained Routing and Scheduling", In M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser (eds), 35-139, Handbooks in Operations Research and Management Science 8: *Network Routing*, Elsevier Science Publishers, Amsterdam, 1995.

J.J. Dongarra, "*Performance of Various Computers Using Standard Linear Equations Software*", Technical Report CS-89-85, Computer Science Department, University of Tennessee, Knoxville, 2008.

G. Dueck and T. Scheurer, "Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing", *Journal of Computational Physics* 90, 161-175 (1990).

B. Funke, T. Grünert and S Irnich, "Local Search for Vehicle Routing and Scheduling Problems: Review and Conceptual Integration", *Journal of Heuristics* 11, 267-306 (2005).

M.R. Garey and D.S. Johnson, *Computers and Intractability. A guide to the theory of NP-Completeness*, W.H. Freeman, New York (1979).

H. Gehring and J. Homberger, "A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows", in *Proceedings of EUROGEN99*, K. Miettinen, M. Mäkelä and J. Toivanen (eds), 57-64, Finland, 1999.

H. Gehring and J. Homberger, "Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows", *Asia-Pacific Journal of Operational Research* 18, 35-47 (2001).

H. Gehring and J. Homberger, "Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows", *Journal of Heuristics* 8, 251-276 (2002).

M. Gendreau, F. Guertin, J-Y. Potvin and R. Séguin, "Neighborhood Search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries", *Transportation Research Part C* 14, 157-174 (2006).

M. Gendreau, A. Hertz and G. Laporte, "A New Insertion and Post-optimization Procedure for the Traveling Salesman Problem", *Operations Research* 40, 1089-1093 (1992).

M. Gendreau, A. Hertz and G. Laporte, "A Tabu Search Heuristic for the Vehicle Routing Problem", *Management Science* 40, 1276-1290 (1994).

M. Gendreau, A. Hertz, G. Laporte and M. Stan, "A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows", *Operations Research* 43, 330-335 (1995).

M. Gendreau, and J-Y. Potvin, "Metaheuristics in Combinatorial Optimization", *Annals of Operations Research* 140, 189-213 (2005).

F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers & Operations Research* 13, 533-549 (1986).

F. Glover, "New Ejection Chain and Alternating Path Methods for Traveling Salesman Problems", in *Computer Science and Operation Research: New Development in their Interfaces*, O. Balci, R. Sharda and S. Zenios (eds), 449-509, Pergamon Press, UK, 1992.

F. Glover, "Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems", *Discrete Applied Mathematics* 65, 223-253 (1996).

B. Golden and A.A. Assad, "Perspectives on Vehicle Routing: Exciting New Developments", *Operations Research* 34, 803-809 (1986).

B.L. Golden, A.A. Assad and E.A.Wasil, "Routing Vehicles in the Real World: Applications in the Solid Waste, Beverage, Food, Dairy, and Newspaper Industries", in *The Vehicle Routing Problem*, P. Toth and D. Vigo (eds), 245-286, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

G. Gutin and Punnen A., *The Traveling Salesman Problem and Its Variations*, Kluwer Academic Publishers, 2002.

W. Harvey and M. Ginsberg, "Limited Discrepancy Search", in *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (IJCAI-95), 607-615, Morgan Kaufmann, 1995.

H. Hashimoto and M. Yagiura, "A path relinking approach with an adaptive mechanism to control parameters for the vehicle routing problem with time windows", in *Lecture Notes in Computer Science*, J. van Hemert and C. Cotta (eds), Vol. 4972, 254-265, EvoCOP 2008, Springer, Berlin, 2008.

H. Hashimoto, M. Yagiura and T. Ibaraki, "An Iterated Local Search for the Time-Dependent Vehicle Routing Problem with Time Windows", *Discrete Optimization* 5, 434-456 (2008).

J. Homberger and H. Gehring, "Two Evolutionary Meta-heuristics for the Vehicle Routing Problem with Time Windows", *INFOR* 37, 297-318 (1999).

J. Homberger and H. Gehring, "A Two-Phase Hybrid Metaheuristic for the Vehicle Routing Problem with Time Windows", *European Journal of Operational Research* 162, 220-238 (2005).

T. Hoshino, T. Kimura and T. Ikeguchi, "Two simple local searches controlled by chaotic dynamics for the vehicle routing problems with time windows", In *Abstract Proceedings of The Seventh Metaheuristics International Conference MIC 2007*, Montreal, Canada, June 25-29, 2007.

T. Ibaraki, S. Imahori, M. Kudo, T. Masuda, T. Uno and M. Yagiura, "Effective Local Search Algorithms for Routing and Scheduling Problems with General Time Window Constraints", *Transportation Science* 39, 206-232 (2005).

T. Ibaraki, S. Imahori, K. Nonobe, K. Sobue, T. Uno and M. Yagiura, "An Iterated Local Search Algorithm for the Vehicle Routing Problem with Convex Time Penalty Functions", *Discrete Applied Mathematics* 156, 2050-2069 (2008).

S. Irnich, "A Unified Modeling and Solution Framework for Vehicle Routing and Local Search-Based Metaheuristics", *INFORMS Journal of Computing* 20(2), 270-287 (2008).

S. Irnich, B. Funke and T. Grónert, "Sequential Search and its Application to Vehicle routing Problems", *Computers & Operations Research* 33, 2405-2429 (2005).

D.E. Joslin and D.P. Clements, "Squeaky Wheel" Optimization", *Journal of Artificial Intelligence Research* 10, 353-373 (1999).

L. Jourdan, M. Basseur and E.-G. Talbi, "Hybridizing exact methods and metaheuristics: A taxonomy", *European Journal of Operational Research* 199, 620-629 (2009).

G. Kindervater and M. Savelsbergh, "Vehicle Routing: Handling Edge Exchanges", in *Local Search in Combinatorial Optimization*, E. Aarts and J.K. Lenstra (eds), 337-360, John Wiley & Sons, Chichester, 1997.

S. Kirkpatrick, C.D. Gelatt and P.M. Vecchi, "Optimization by Simulated Annealing", *Science* 220, 671-680 (1983).

J. Kytöjoki, T. Nuortio, O. Bräysy and M. Gendreau, "An Efficient Variable Neighborhood Search Heuristic for Very Large Scale Vehicle Routing Problems", *Computers & Operations Research* 34, 2743-2757 (2007).

A. Le Bouthillier and T.G. Crainic, "Co-operative Parallel Metaheuristic for Vehicle Routing with Time Windows", *Computers & Operations Research* 32, 1685-1708 (2005).

A. Le Bouthillier, T.G. Crainic and P. Kropf, "A Guided Cooperative Search for the Vehicle Routing Problem with Time Windows", *IEEE Intelligent Systems* 20(4), 36-42 (2005).

F. Li, B. Golden and E. Wasil, "Very Large-scale Vehicle Routing: New Test Problems, Algorithms, and Results", *Computers & Operations Research* 32, 1165-1179 (2005).

F. Li, B. Golden and E. Wasil, "The Open Vehicle Routing Problem: Algorithms, Large-scale Test Problems, and Computational Results", *Computers & Operations Research* 34, 2918-2930 (2007).

F. Li, B. Golden and E. Wasil, "A Record-to-Record Travel Algorithm for Solving the Heterogeneous Fleet Vehicle Routing Problem", *Computers & Operations Research* 34, 2734-2742 (2007).

A. Lim and X. Zhang, "A Two-stage Heuristic with Ejection Pools and Generalized Ejection Chains for the Vehicle Routing Problem with Time Windows", *INFORMS Journal on Computing* 19, 443-457 (2007).

H.R. Lourenço, O. Martin and T. Stützle, "Iterated Local Search", in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger (eds), 321-353, International Series in Operations Research and Management Science, Kluwer Academic Publishers, Norwell, 2002.

D. Mester and O. Bräysy, "Active Guided Evolution Strategies for Large Scale Vehicle Routing Problems with Time Windows", *Computers & Operations Research* 32, 1593-1614 (2005).

D. Mester, O. Bräysy and W. Dullaert, "A Multi-parametric Evolution Strategies Algorithm for Vehicle Routing Problems", *Expert Systems with Applications* 32, 508-517 (2006).

N. Mladenović and P. Hansen, "Variable Neighborhood Search", *Computers & Operations Research* 24, 1097-1100 (1997).

Y. Nagata and O. Bräysy, "A powerful route minimization heuristic for the vehicle routing problem with time windows", *Operations Research Letters* 37, 333-338 (2009a).

Y. Nagata and O. Bräysy, "Edge Assembly based Memetic Algorithm for the Capacitated Vehicle Routing Problem", *Networks* 54, 205-215 (2009b).

Y. Nagata, O. Bräysy and W. Dullaert, "A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows", *Computers & Operations Research* 37, 724-737 (2010).

I. Or, Traveling "*Salesman-type Combinatorial Problems and their Relation to the Logistics of the Regional Blood Banking*", PhD Thesis, Northwestern University, IL (1976).

I.H. Osman, "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem", *Annals of Operations Research* 41, 421-452 (1993).

I.H. Osman and G. Laporte, "Metaheuristics: A Bibliography", *Annals of Operations Research* 63, 513-623 (1996).

G. Pesant and M. Gendreau, "A Constraint Programming Framework for Local Search Methods", *Journal of Heuristics* 5, 255-279 (1999).

D. Pisinger and S. Ropke, "A General Heuristic for Vehicle Routing Problems", *Computers & Operations Research* 34, 2403-2435 (2006).

J.-Y. Potvin, G. Lapalme and J.M. Rousseau, "A generalized k-opt exchange procedure for the MTSP", *Information Systems and Operational Research* 27(4), 474-481 (1989).

J.-Y. Potvin and J.M. Rousseau, "A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows", *European Journal of Operational Research* 66, 331-340 (1993).

J.-Y. Potvin and J.M. Rousseau, "An Exchange Heuristic for Routeing Problems with Time Windows", *Journal of Operational Research Society* 46, 1433-1446 (1995).

E. Prescott-Gagnon, G. Desaulniers, and L.-M. Rousseau, "A Branch-and-Brice Large Neighborhood Search Algorithm for the Vehicle Routing Problem with Time Windows", *Networks* 54 (4), 190-204, (2009).

P.P. Repoussis, C.D. Tarantilis and G. Ioannou, "An Arc-Guided Evolutionary Algorithm for the Vehicle Routing Problem with Time Windows", *IEEE Transactions on Evolutionary Computation* 13(3), 624-647 (2009).

Y. Rochat and E.D. Taillard, "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing", *Journal of Heuristics* 1, 147-167 (1995).

L.-M. Rousseau, M. Gendreau and G. Pesant, "Using Constraint-Based Operators to Solve the Vehicle Routing Problem with Time Windows", *Journal of Heuristics* 8, 43-58 (2002).

R.A. Russell, "Hybrid Heuristics for the Vehicle Routing Problem with Time Windows", *Transportation Science* 29, 156-166 (1995).

S. Sahoo, S. Kim, B.-I. Kim, B. Kraas, A. Popov Jr., "Routing Optimization for Waste Management", *Interfaces* 35, 24-36 (2005).

M.W.P. Savelsbergh, "Local Search in Routing Problems with Time Windows", *Annals of Operations Research* 4, 285-305 (1985).

M.W.P. Savelsbergh, "The Vehicle Routing Problem with Time Windows: Minimizing Route Duration", *ORSA Journal on Computing* 4, 146-154 (1992).

P. Shaw, "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems", in *Lecture Notes in Computer Science*, M. Maher and J.-F. Puget (eds), Vol. 1520, 417-431, Principles and Practice of Constraint Programming - CP98, Springer, Berlin, 1998.

M.M. Solomon, "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints", *Operations Research* 35, 254-265 (1987).

H. Sontrop, P. van der Horn and M. Uetz, "Fast Ejection Chain Algorithm for Vehicle Routing with Time Windows", in *Lecture Notes in Computer Science*, M.J. Blesa et al. (eds), Vol. 3636, 78-89, Hybrid Metaheuristics - HM2005, Springer, Berlin, 2005.

E.D. Taillard, P. Badeau, M. Gendreau, F. Guertin and J.-Y. Potvin, "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows", *Transportation Science* 31, 170-186 (1997).

E. Talbi, "A Taxonomy of Hybrid Metaheuristics", *Journal of Heuristics* 8, 541-564 (2002).

P.M. Thompson and H.N. Psaraftis, "Cyclic Transfer Algorithms for Multivehicle Routing and Scheduling Problems", *Operations Research* 41, 935-946 (1993).

P. Toth and D. Vigo, "The Granular Tabu Search and its Application to the Vehicle Routing Problem", *INFORMS Journal on Computing* 15(4), 333-346 (2003).

C. Voudouris and E. Tsang, "Guided Local Search", *European Journal of Operational Research* 113(2), 469-499 (1999).

D. Weigel and B. Cao, "Applying GIS and OR Techniques to Solve Sears Technician-dispatching and Home-delivery Problems", *Interfaces* 29(1), 112-130 (1999).